



National Aeronautics and  
Space Administration

**Jet Propulsion Laboratory**  
California Institute of Technology  
Pasadena, California

# Introduction to System Modeling and Ontologies

**Steven Jenkins**  
**Engineering Development Office**  
**Systems and Software Division**



Copyright © 2011 California Institute of Technology. Government sponsorship acknowledged.  
This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology,  
under a contract with the National Aeronautics and Space Administration.



National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Agenda

31 Systems + Software

- **Motivating example**
- **Reasoning about models**
- **Some objectives of modeling**
- **Presentations and facts**
- **Ontologies and facts**
- **JPL/IMCE ontologies**
- **Ontology standards**
- **Ontologies and SysML**
- **Closing thoughts**

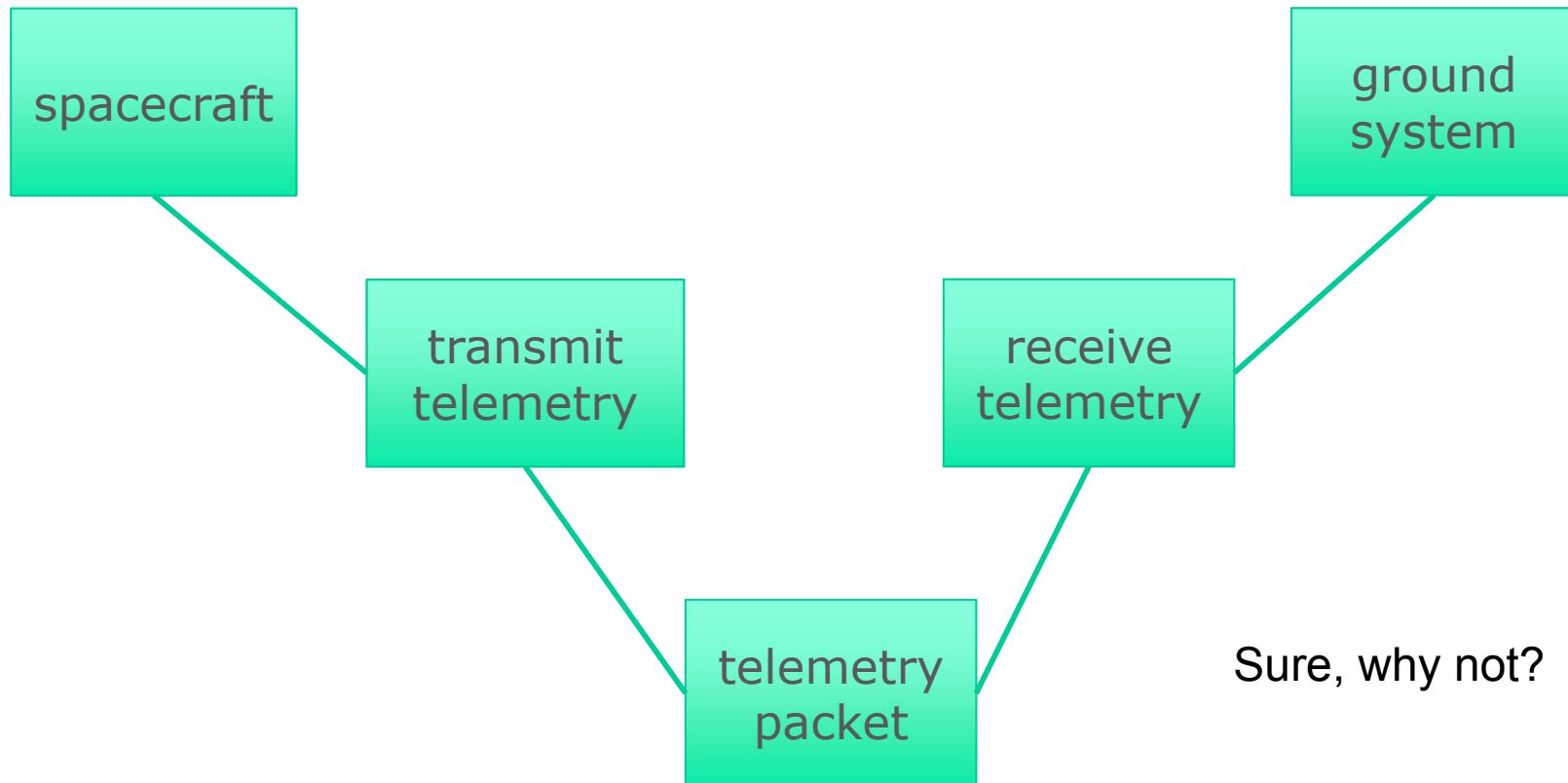


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Is This A Model?

31 Systems + Software



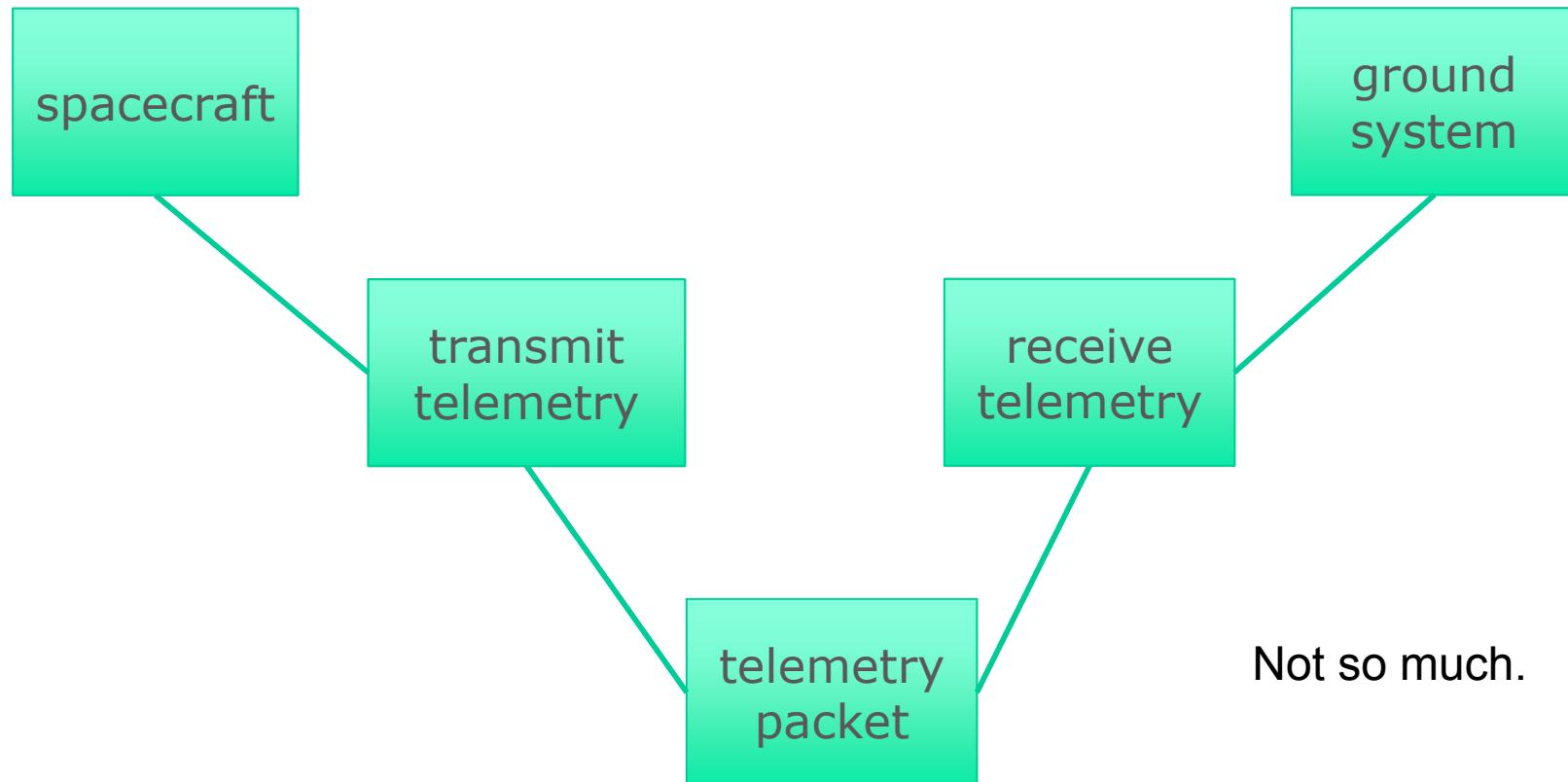


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Is It A Good Model?

31 Systems + Software



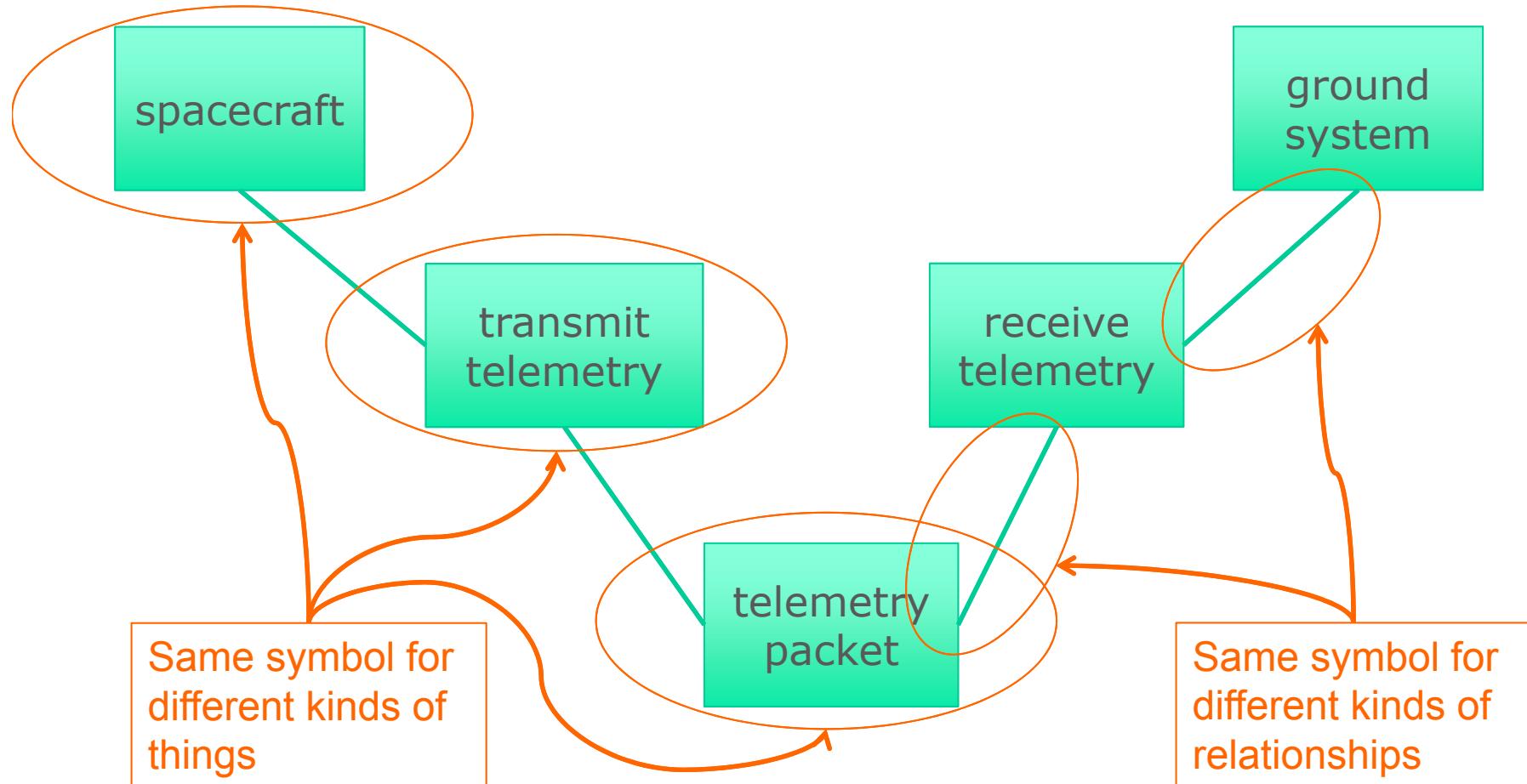


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# What's Wrong With It?

31 Systems + Software



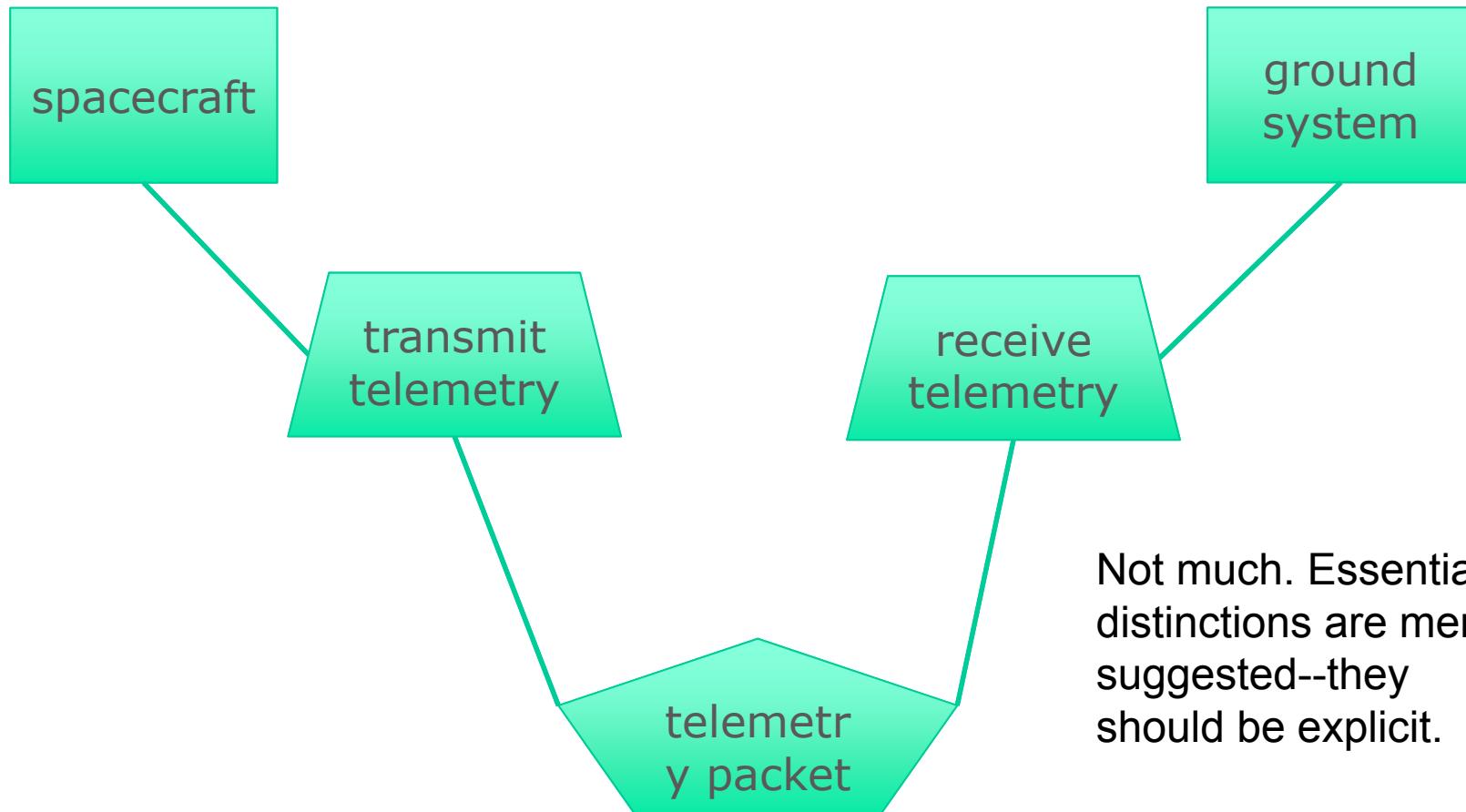


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Better?

31 Systems + Software



Not much. Essential distinctions are merely suggested--they should be explicit.



# Making Distinctions Explicit

31 Systems + Software

- **Rather than merely hinting at distinctions with shapes or colors, we could devise a set of types or classes to be applied to model elements**
- **The set of types is application-dependent**
  - **Systems engineers talk about different things from chefs**
  - **The distinctions are whatever matters for your application**
  - **Is red wine a different type from white, or is it merely a property of wine?**
    - It depends on what you want to say about wine
- **What kinds of things do systems engineers talk about?**
  - **Component, Interface, Function, Requirement, Work Package, Product, Process, Objective, Message, etc.**
- **Let's apply some classes to our model**
- **For now, every element has**
  - **one type, denoted like this: «type»**
  - **one name, which identifies an individual of that type**

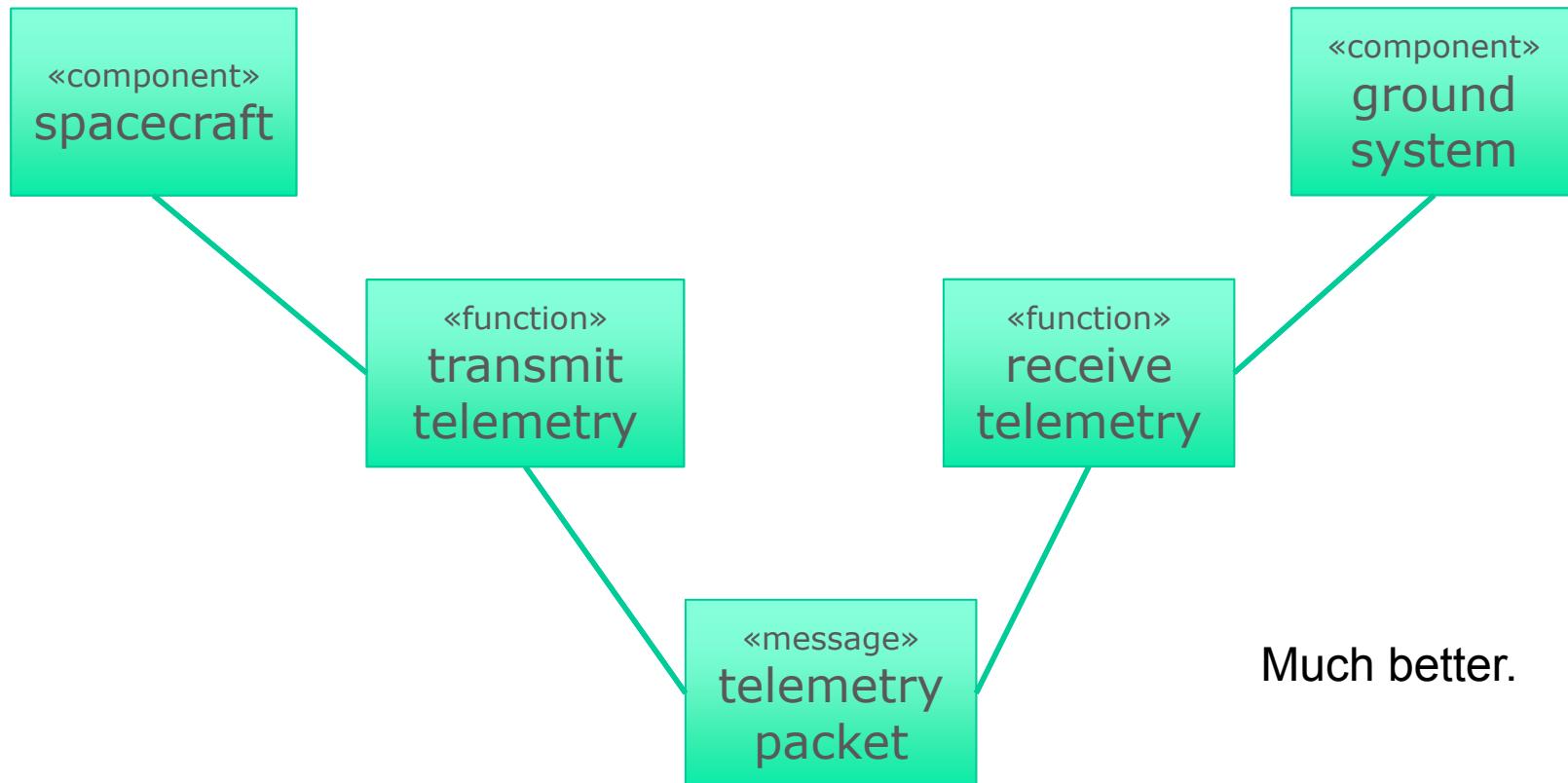


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Model With Typed Elements

31 Systems + Software



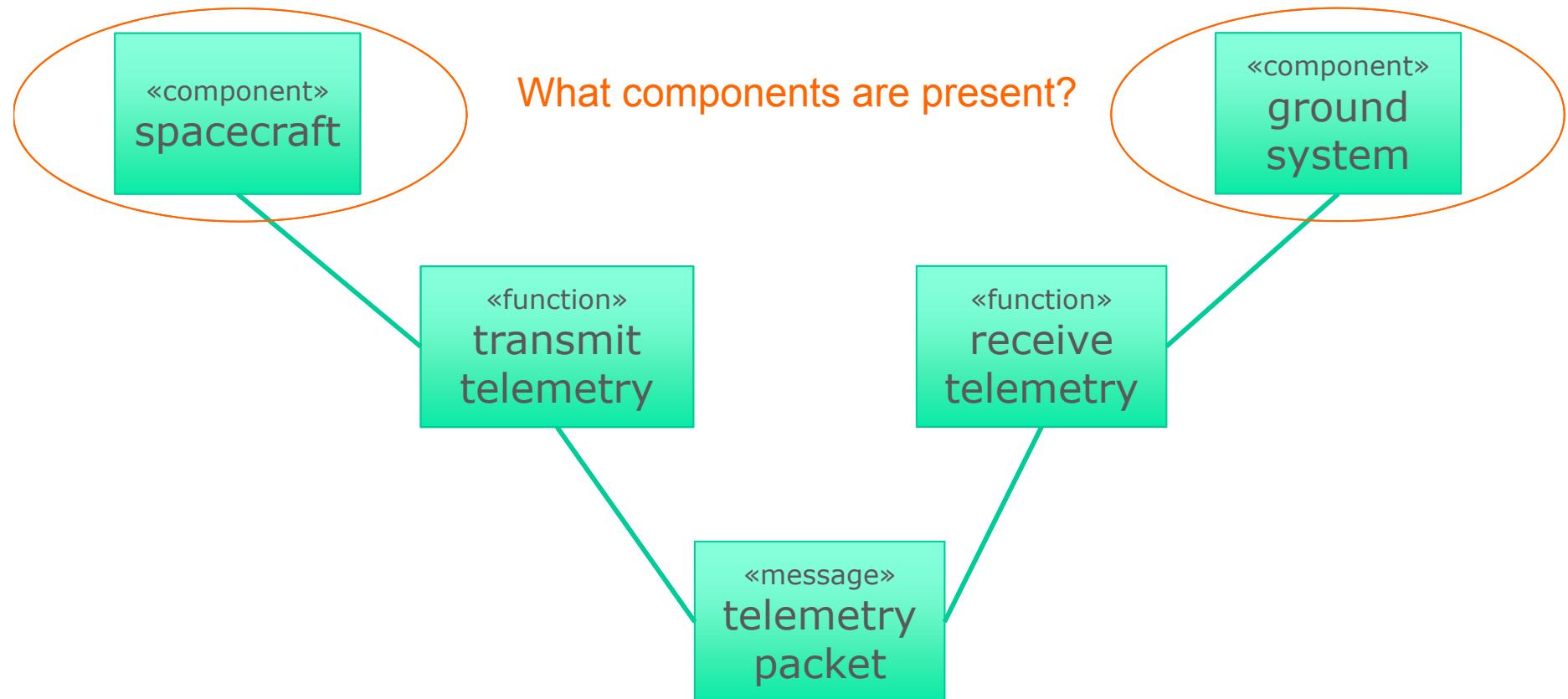


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Answering Questions

31 Systems + Software



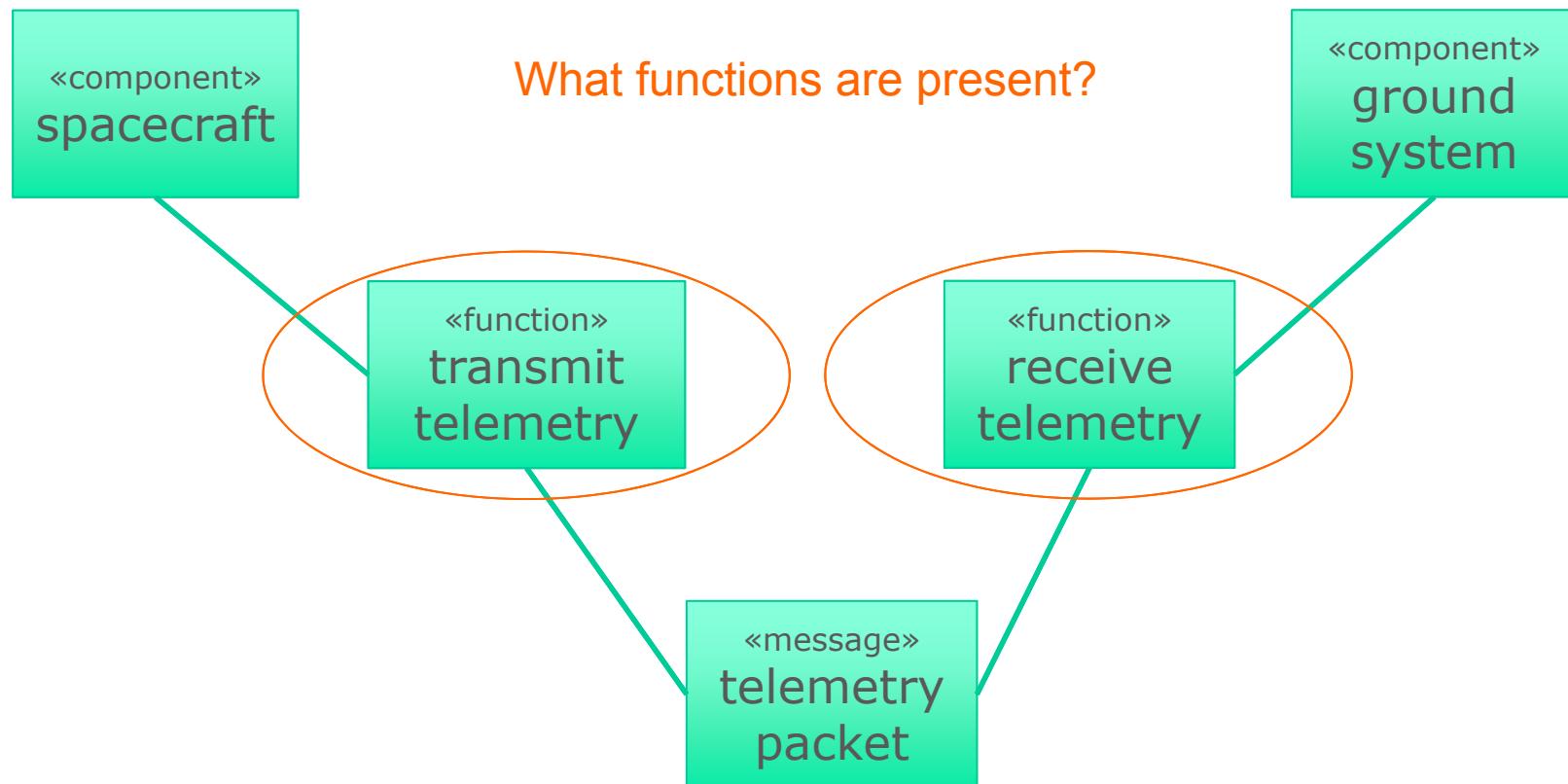


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Answering Questions

31 Systems + Software



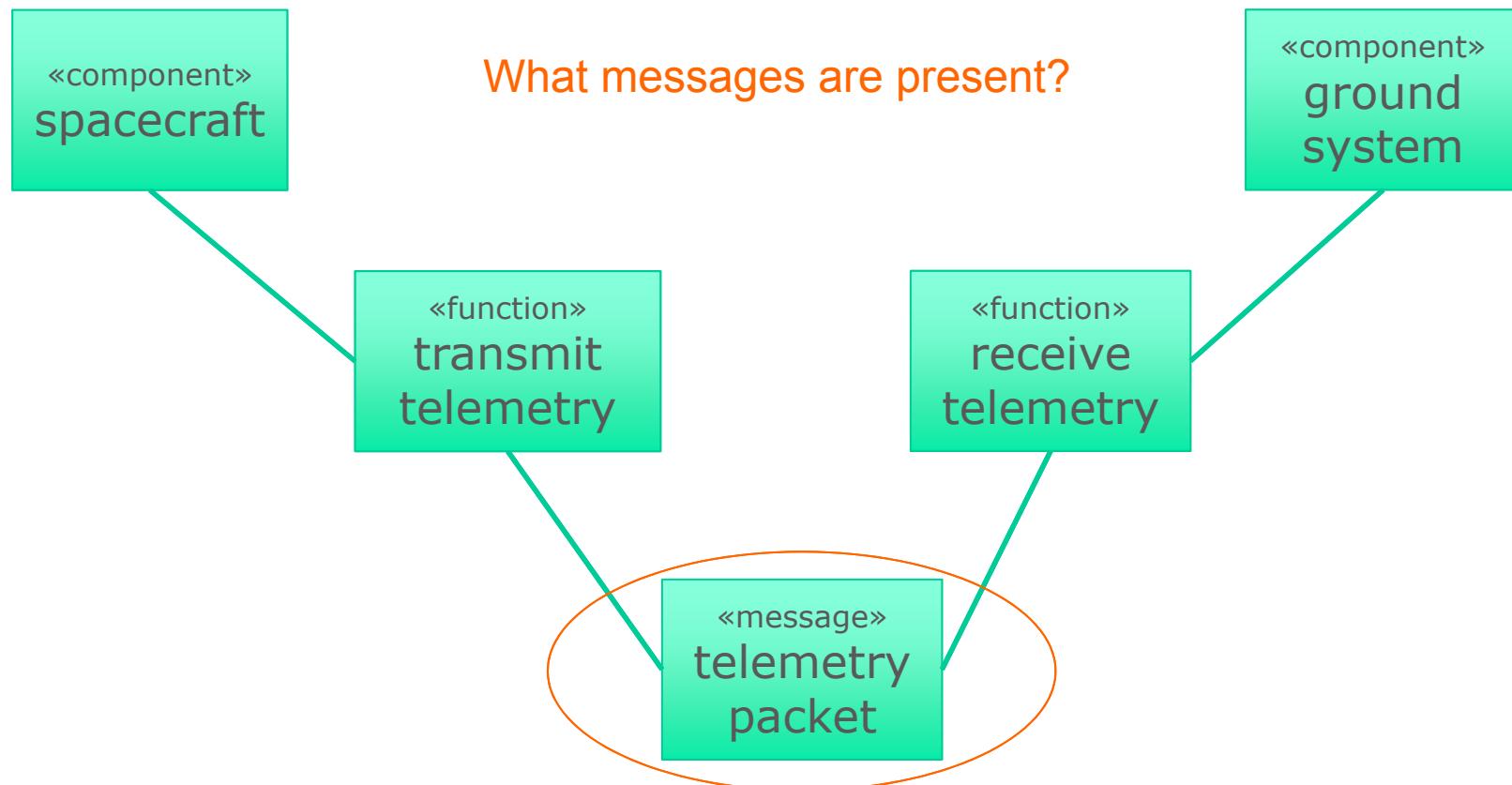


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Answering Questions

31 Systems + Software



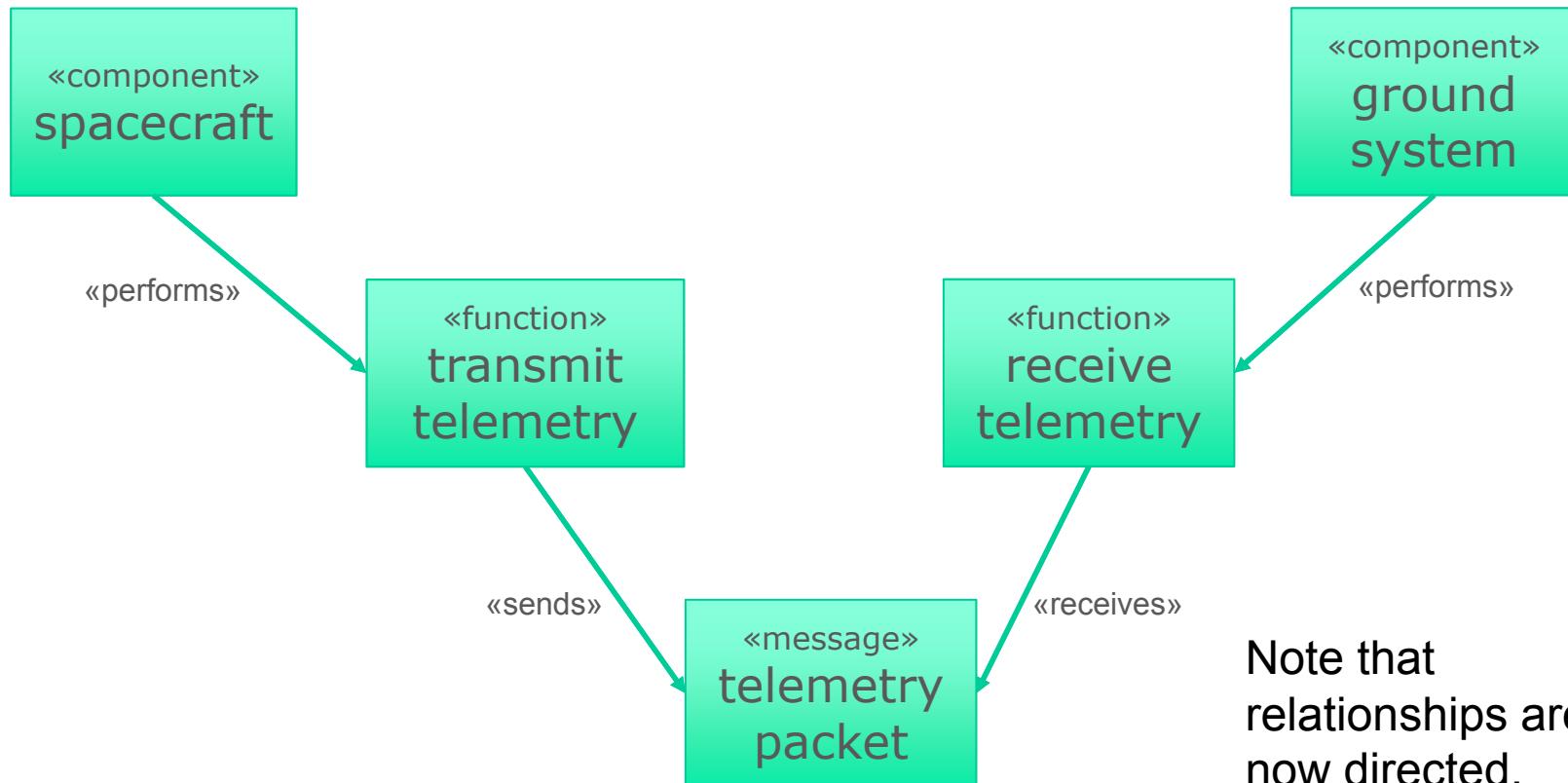


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

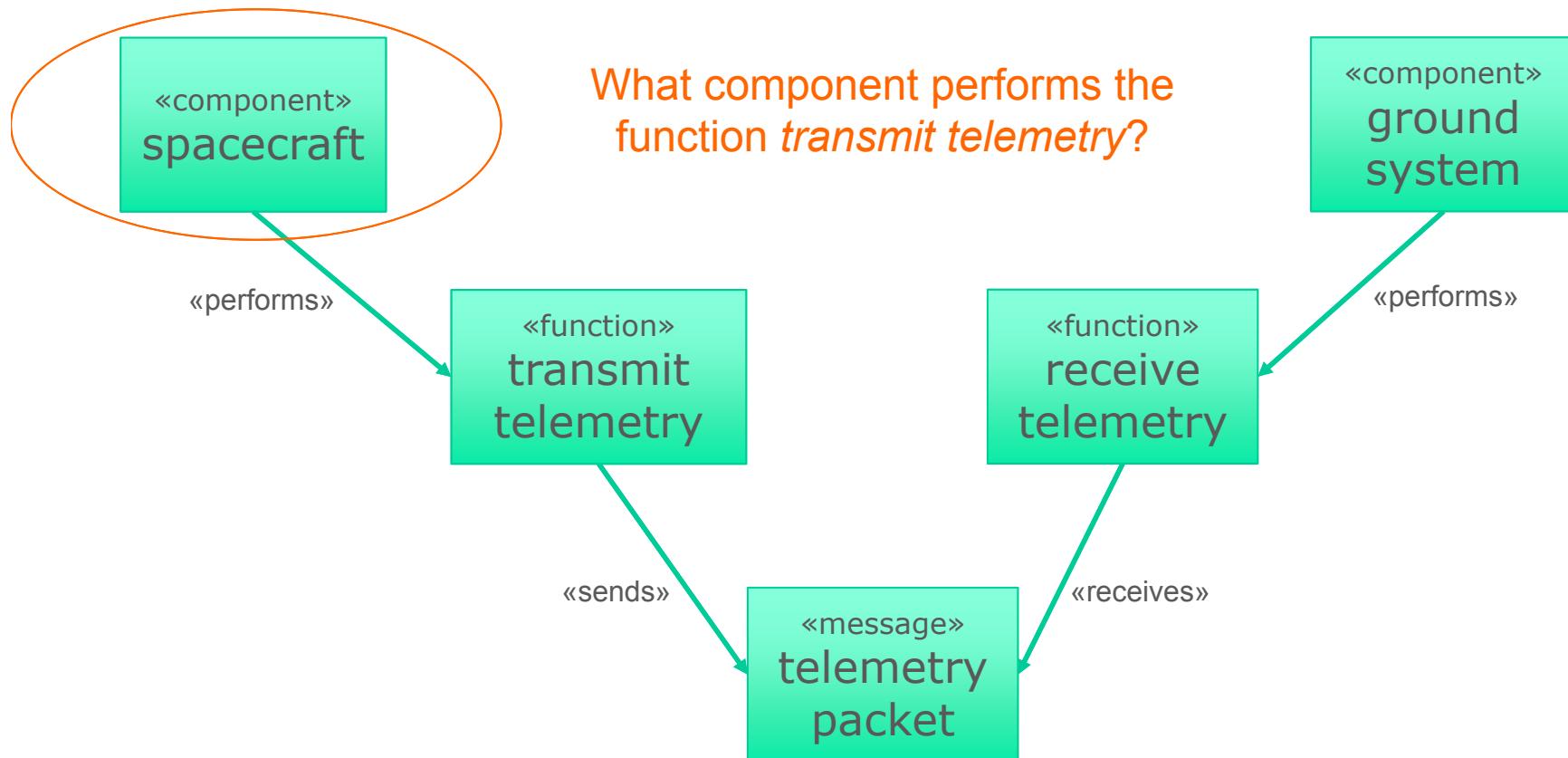
# Add Typed Relationships

31 Systems + Software





## More Questions and Answers



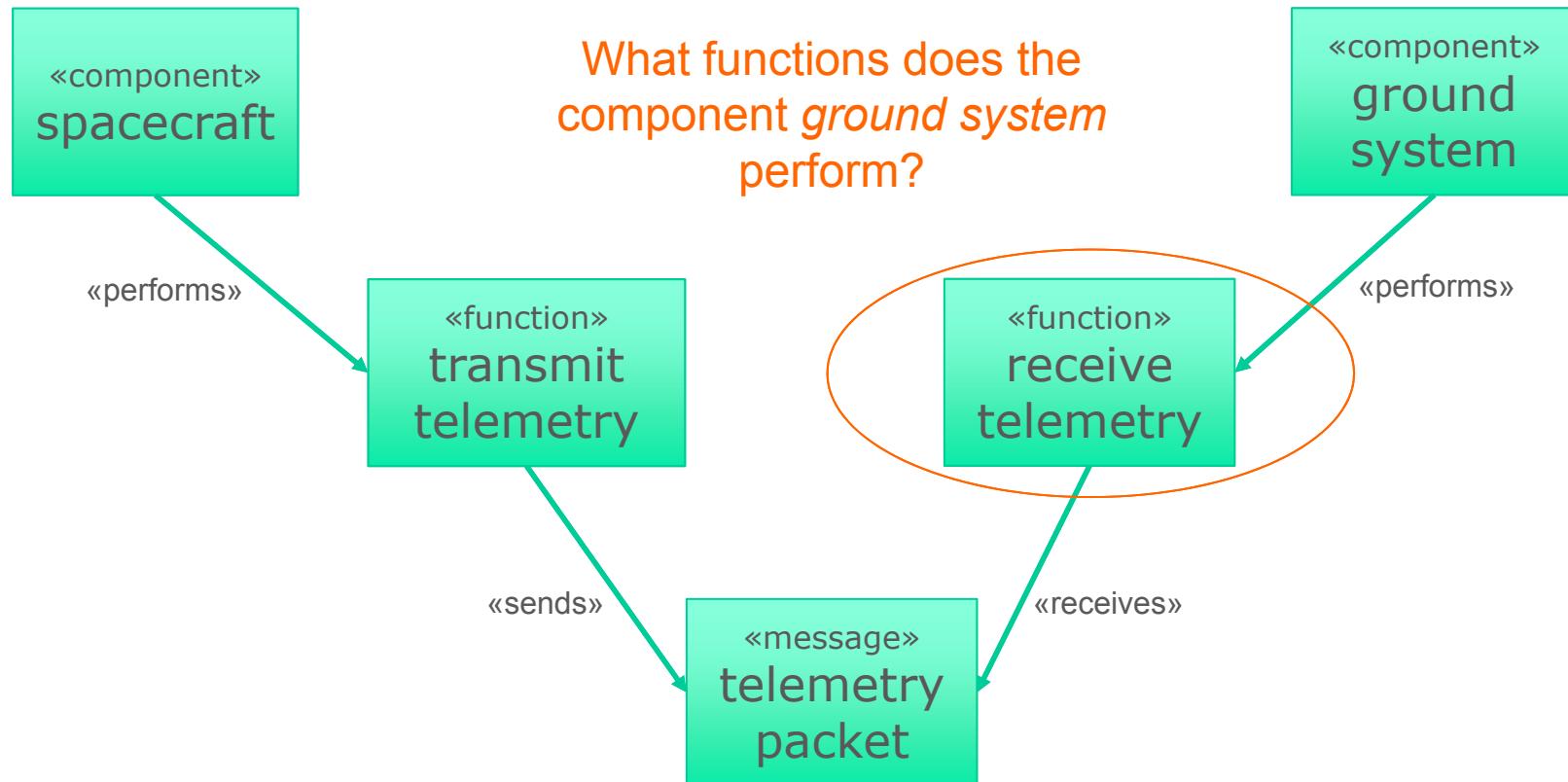


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

## More Questions and Answers

31 Systems + Software



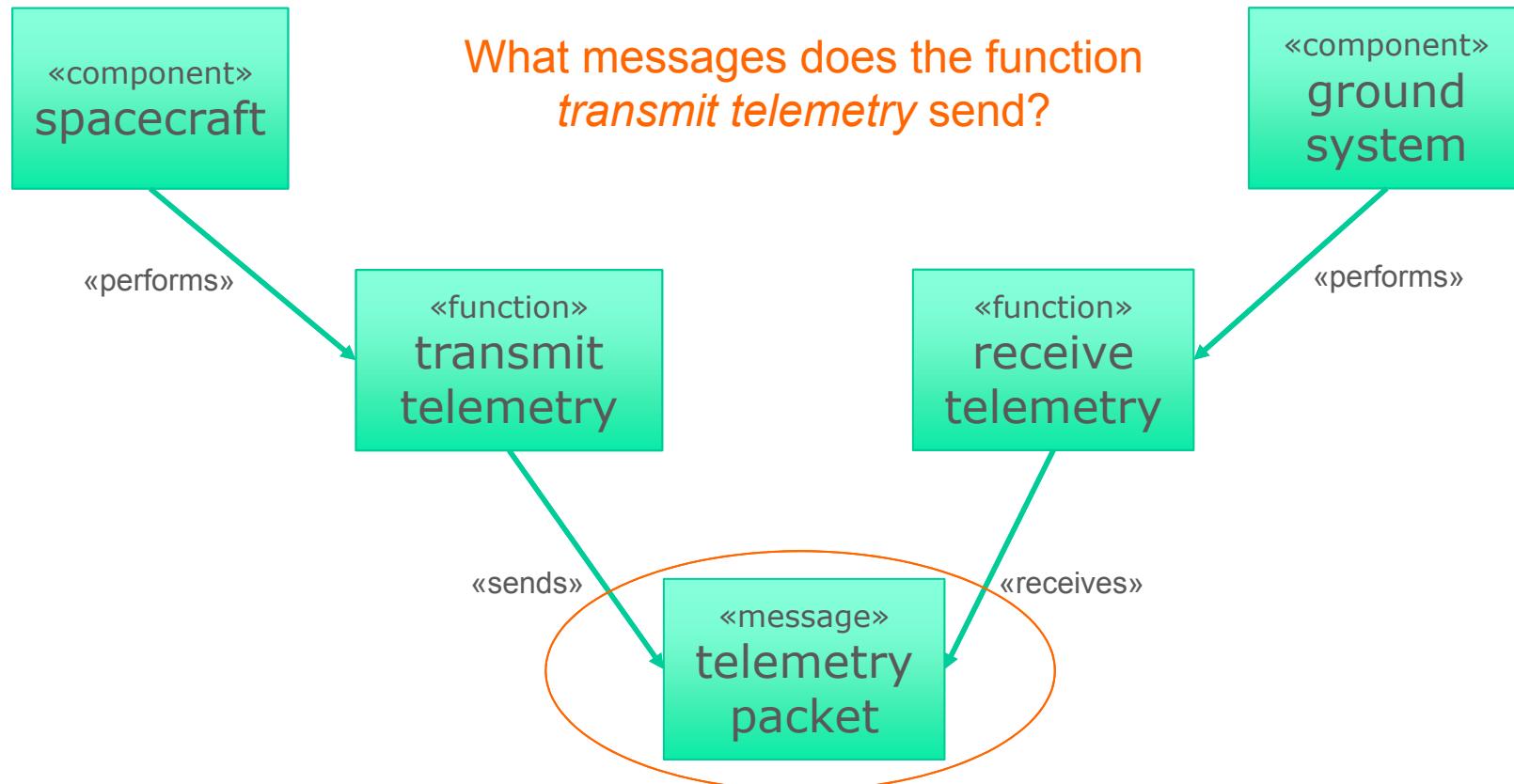


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# More Questions and Answers

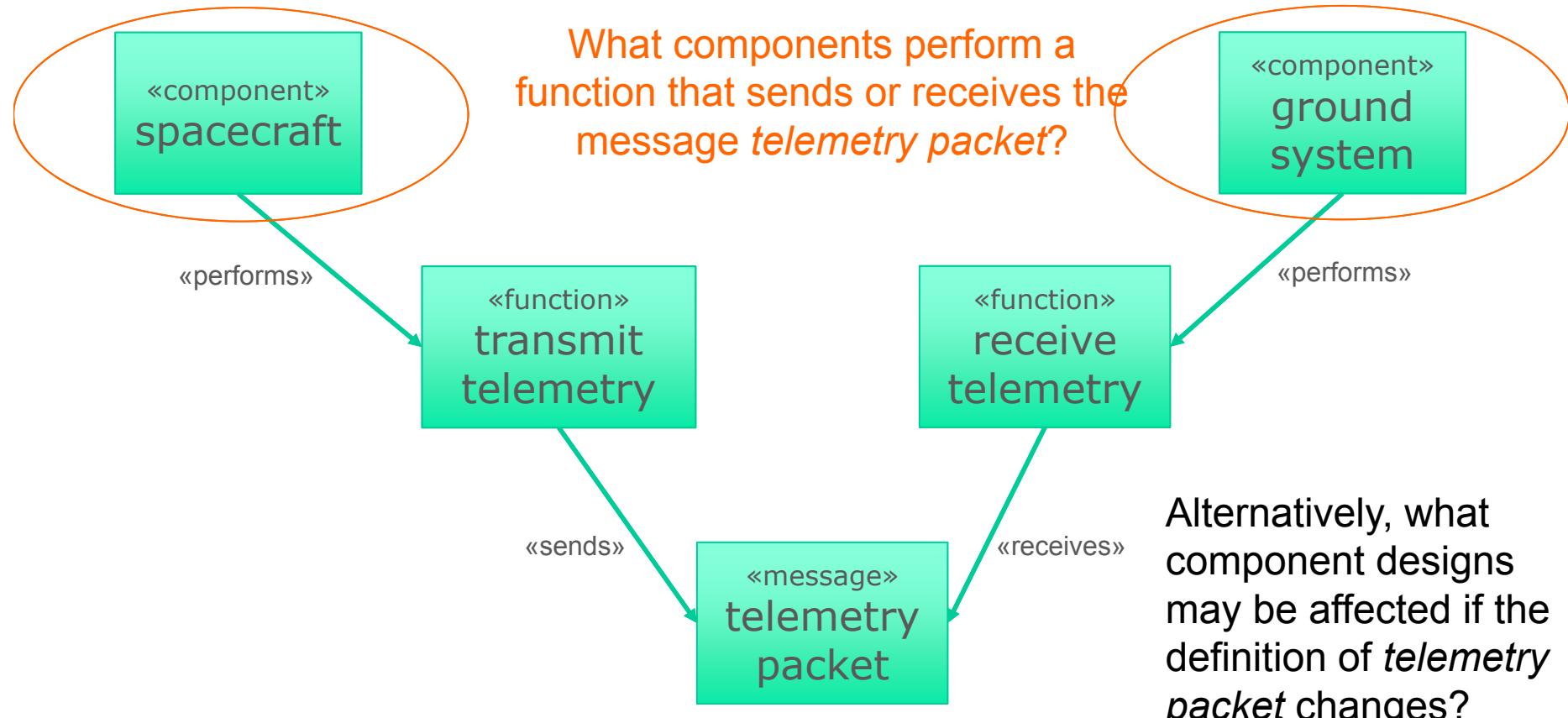
31 Systems + Software





## More Questions and Answers

31 Systems + Software





National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Reasoning About Models

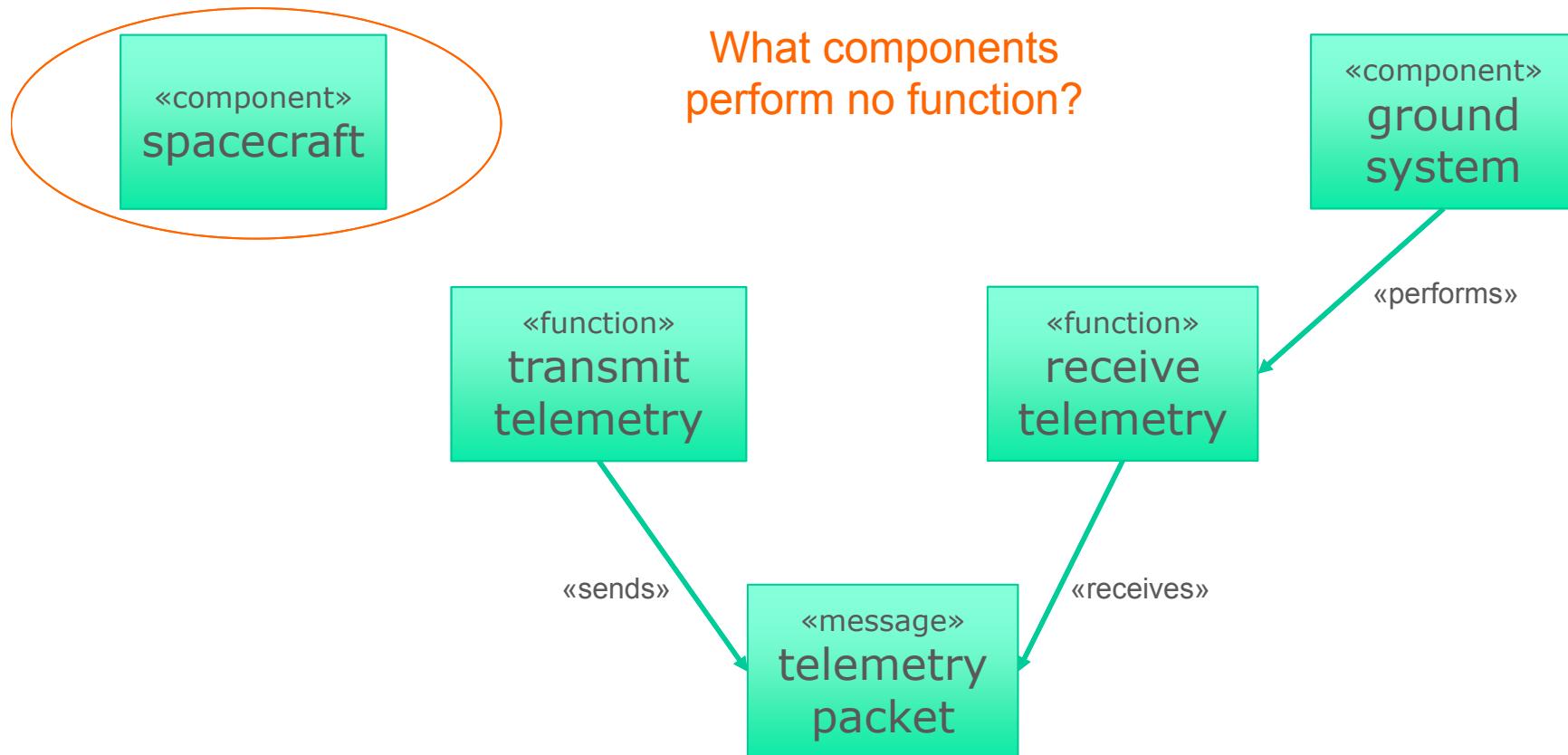
31 Systems + Software

- **We can use models to answer questions**
- **The questions may be about the system itself**
  - **What is it?**
  - **How does it work?**
  - **Is the performance adequate?**
  - **What happens if something breaks?**
- **The questions may be about the model**
  - **Is it complete?**
  - **Is it consistent?**
  - **Does it support required analyses?**
- **The questions may be about the design artifacts**
  - **Are all required documents present?**
  - **Does each document contain all required content?**
- **We call answering these kinds of questions *reasoning***
  - It doesn't necessarily mean exotic, artificial intelligence



# Reasoning About Completeness

31 Systems + Software



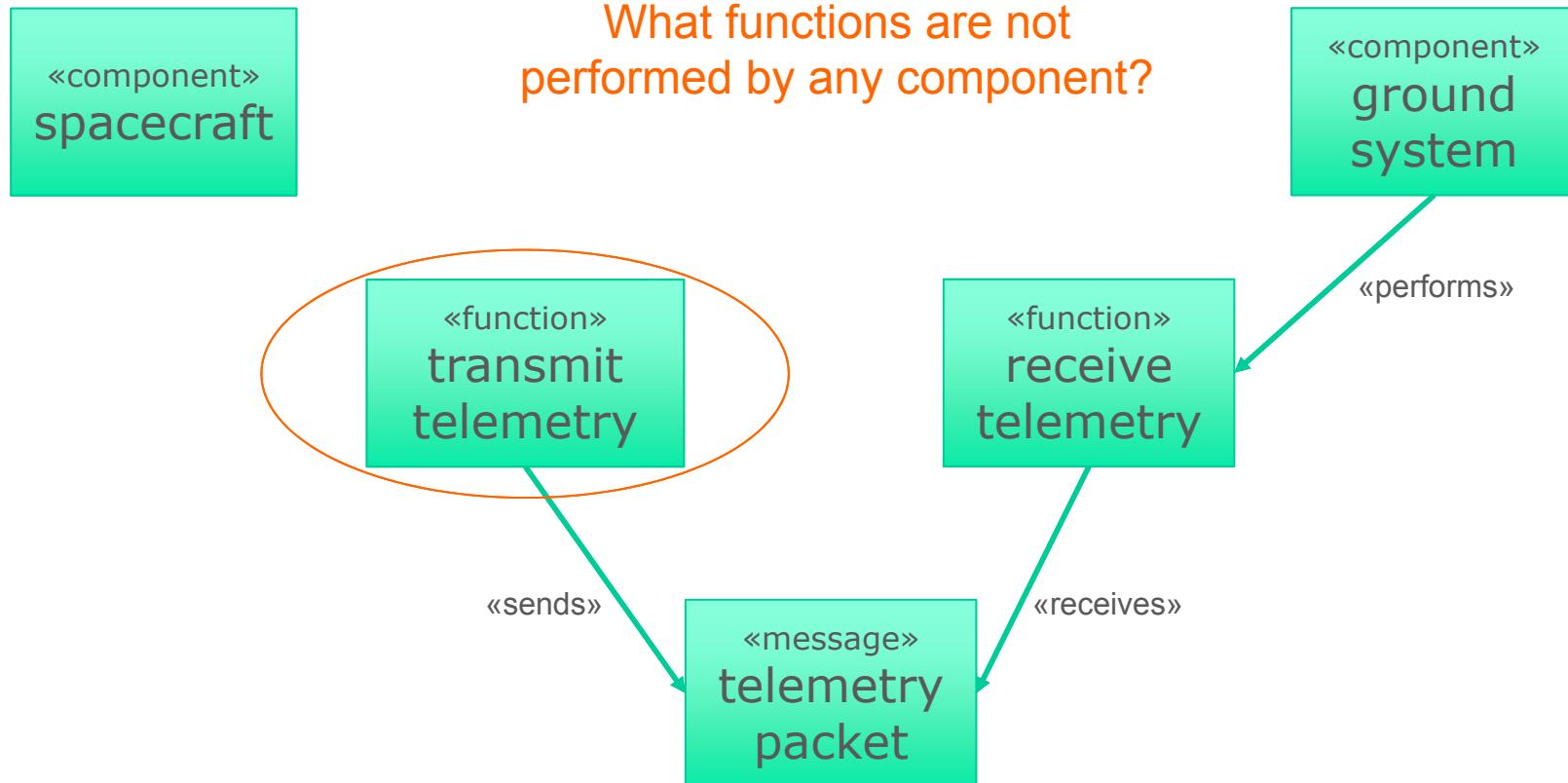


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Reasoning About Completeness

31 Systems + Software



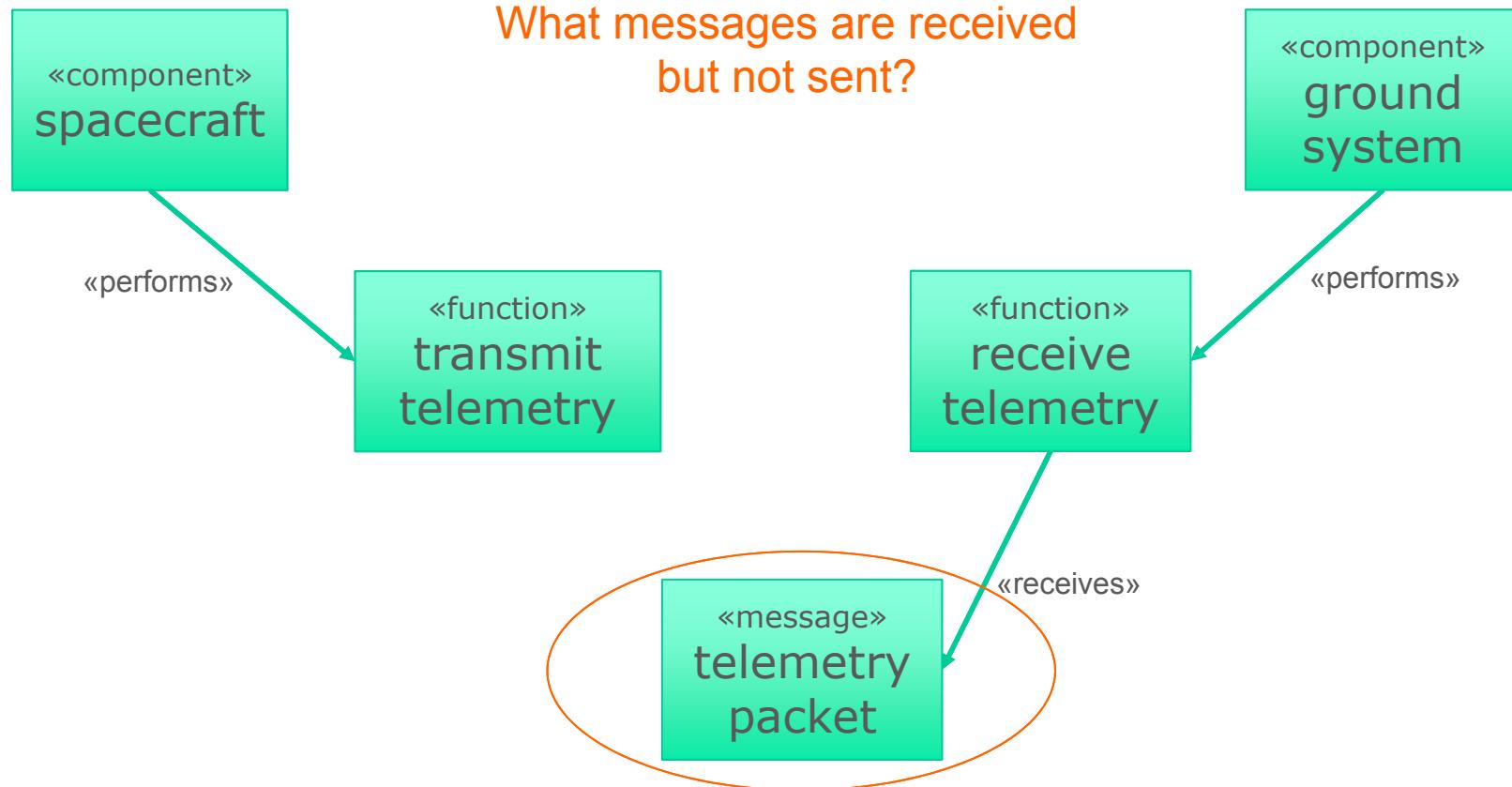


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Reasoning About Completeness

31 Systems + Software



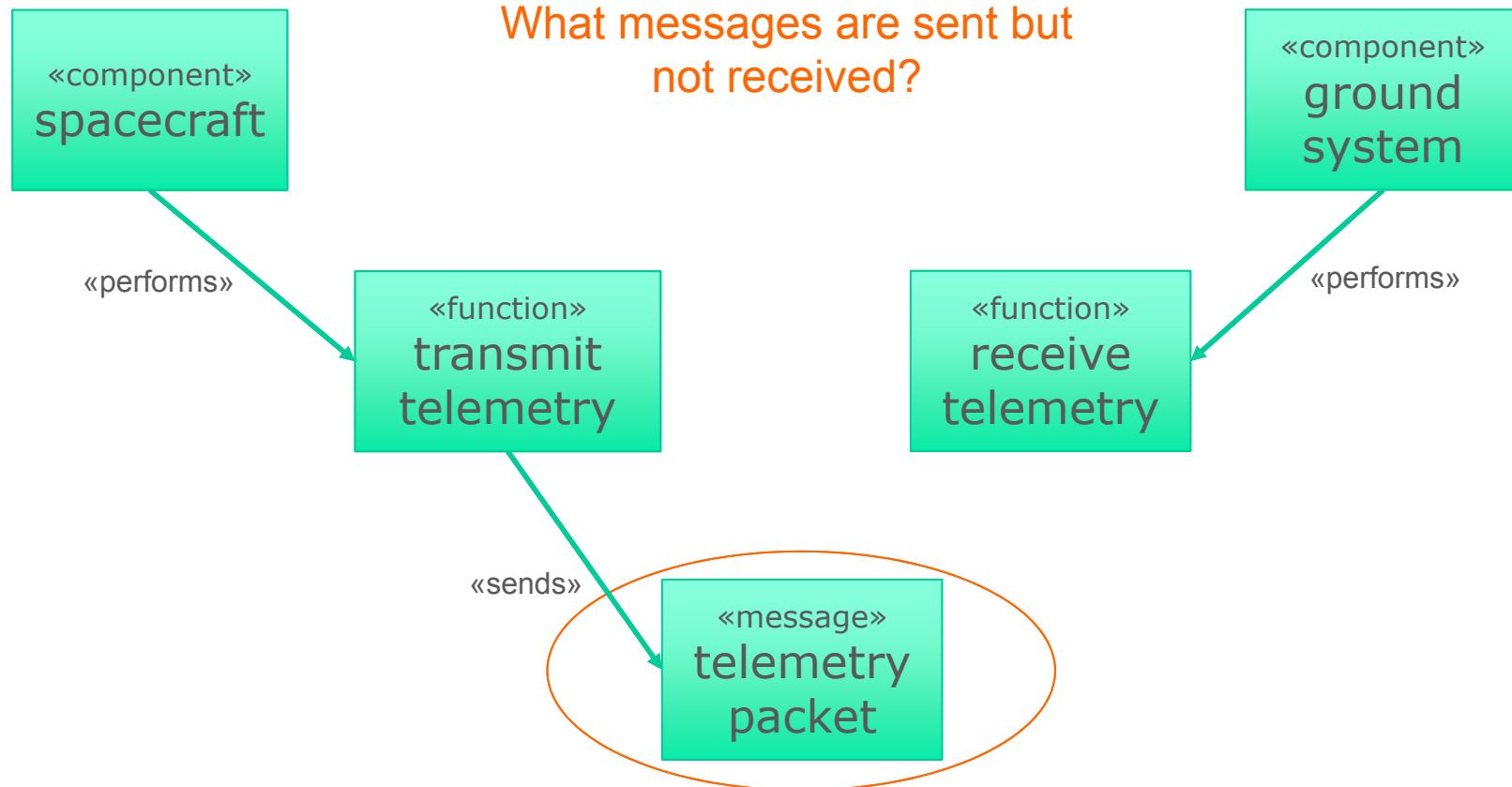


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Reasoning About Completeness

31 Systems + Software



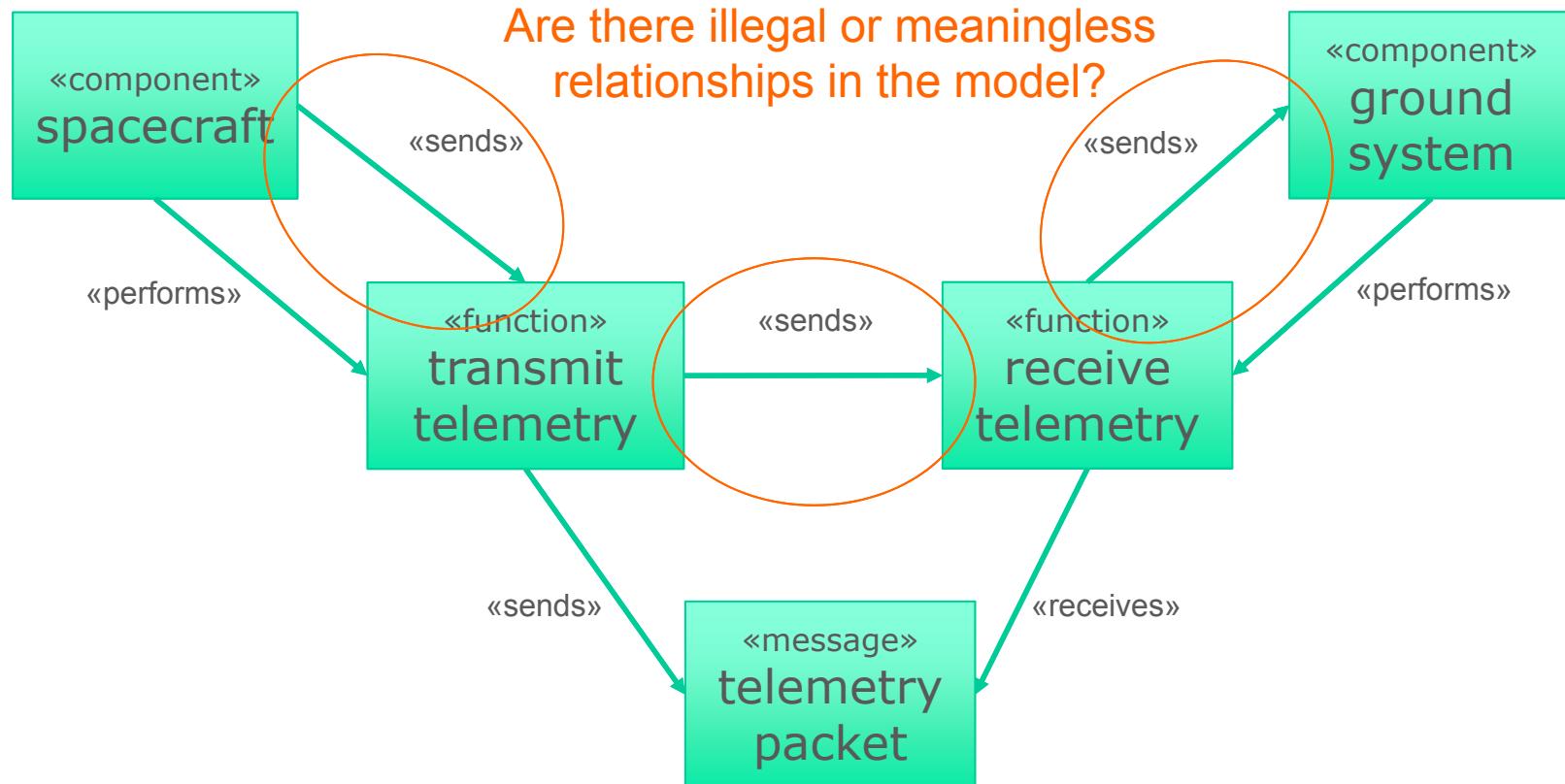


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Reasoning About Consistency

31 Systems + Software



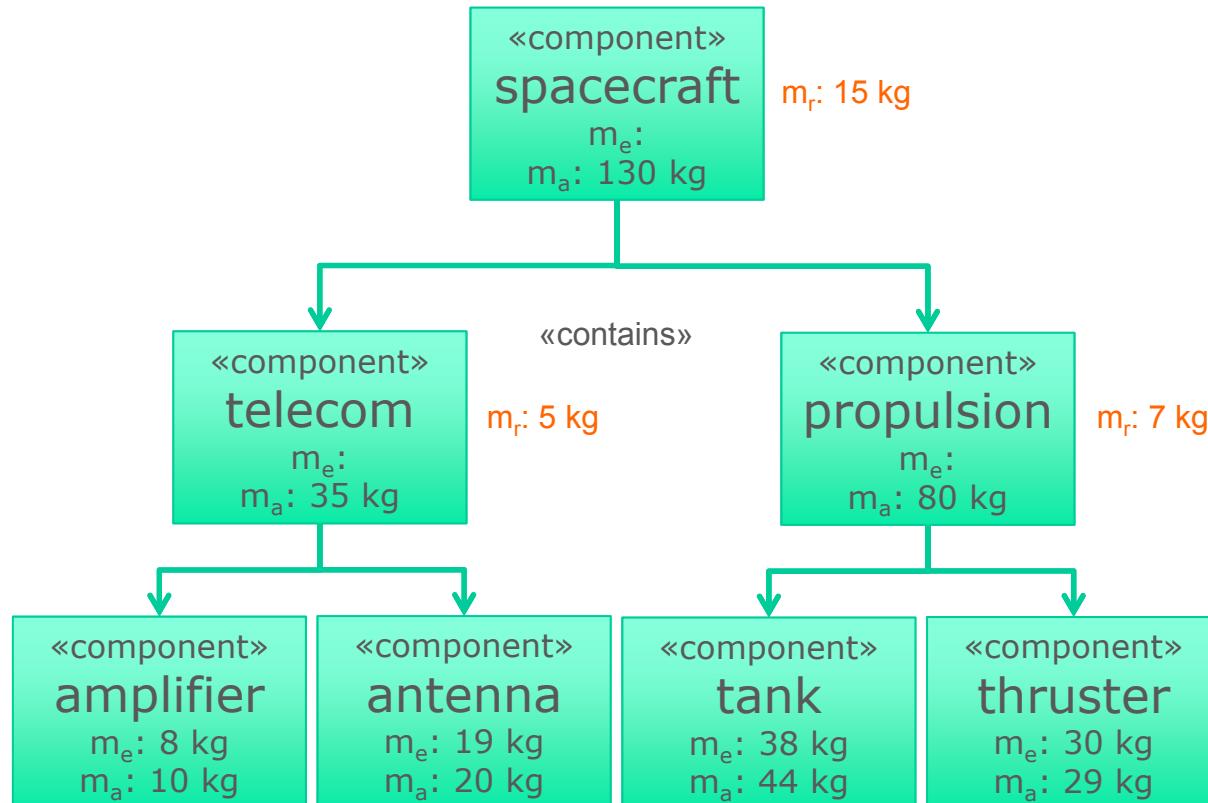


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Reasoning About Design

31 Systems + Software



Rule: Reserve mass  $m_r$  of any component with parts is the difference between its  $m_a$  and the sum of  $m_a$  of its parts

$m_e$ : estimated mass  
 $m_a$ : allocated mass

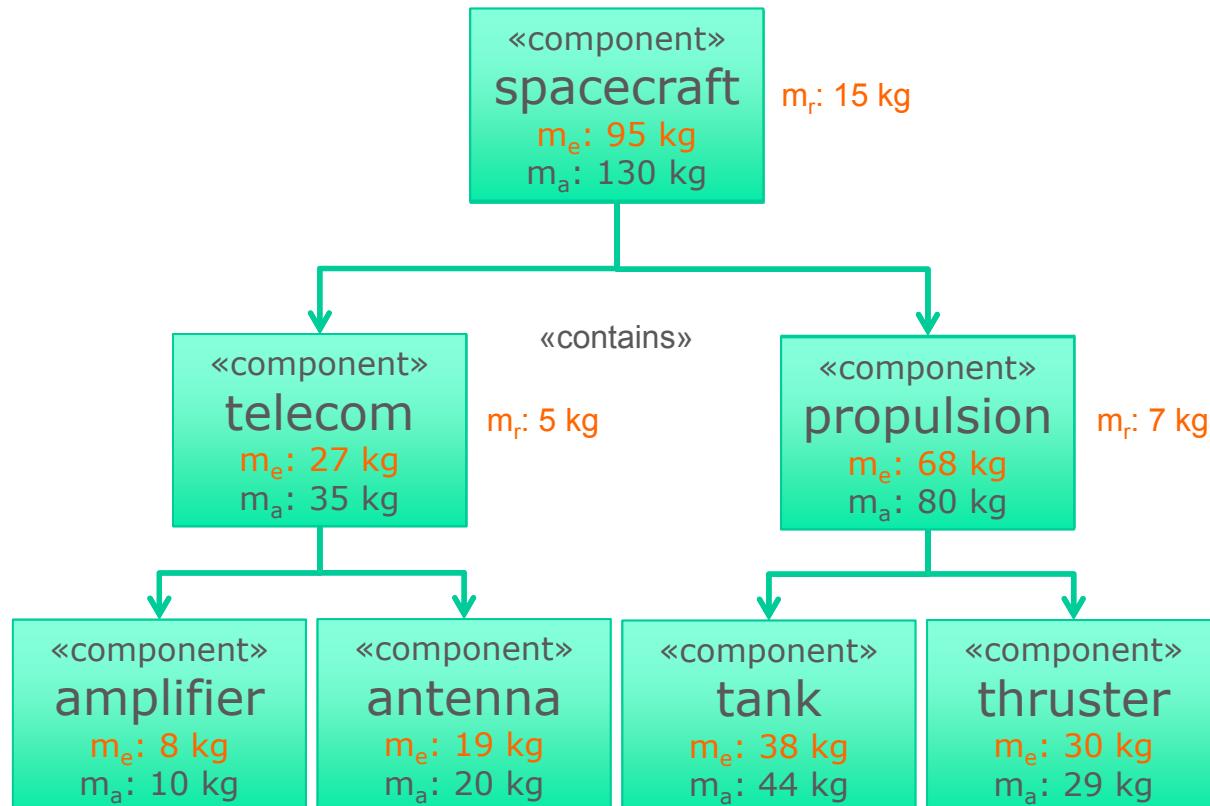


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Reasoning About Design

31 Systems + Software



Rule: Reserve mass  $m_r$  of any component with parts is the difference between its  $m_a$  and the sum of  $m_a$  of its parts

Rule: CBE mass  $m_e$  of any component with parts is the sum of  $m_e$  of its parts

$m_e$ : estimated mass  
 $m_a$ : allocated mass

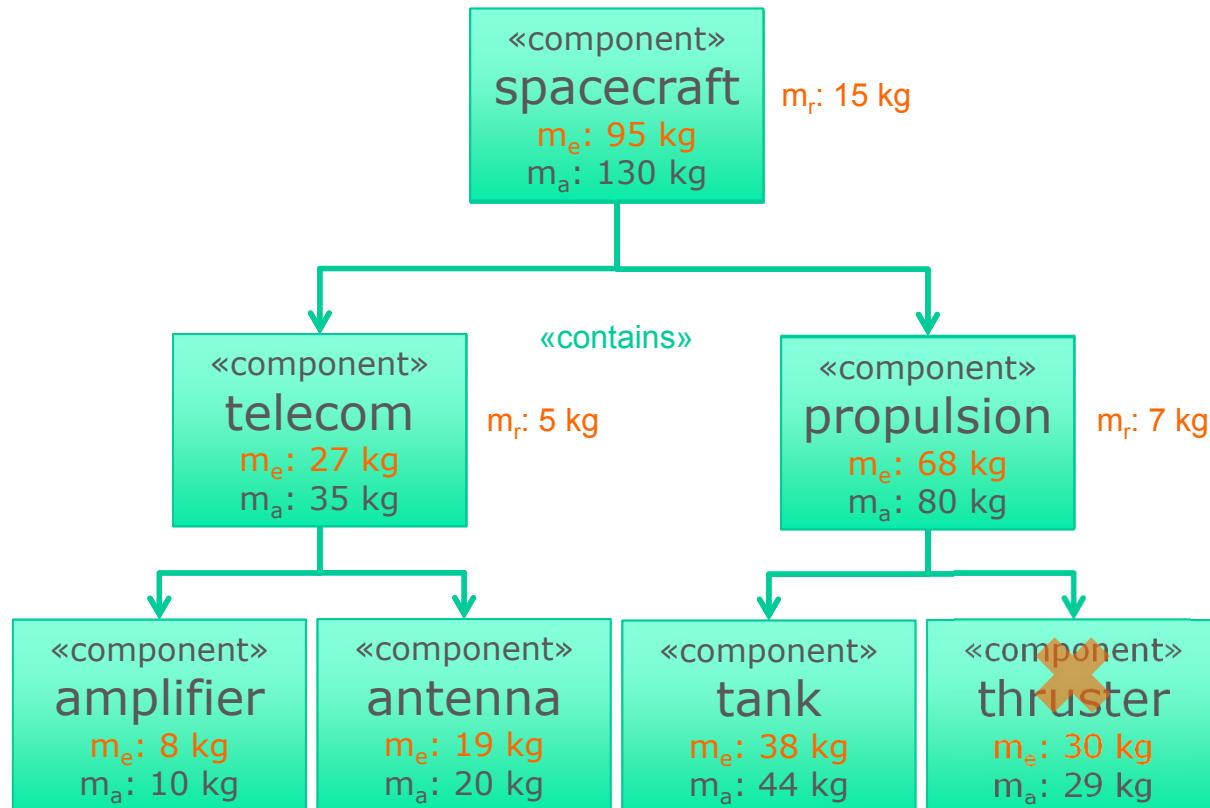


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Reasoning About Design

31 Systems + Software



Rule: Reserve mass  $m_r$  of any component with parts is the difference between its  $m_a$  and the sum of  $m_a$  of its parts

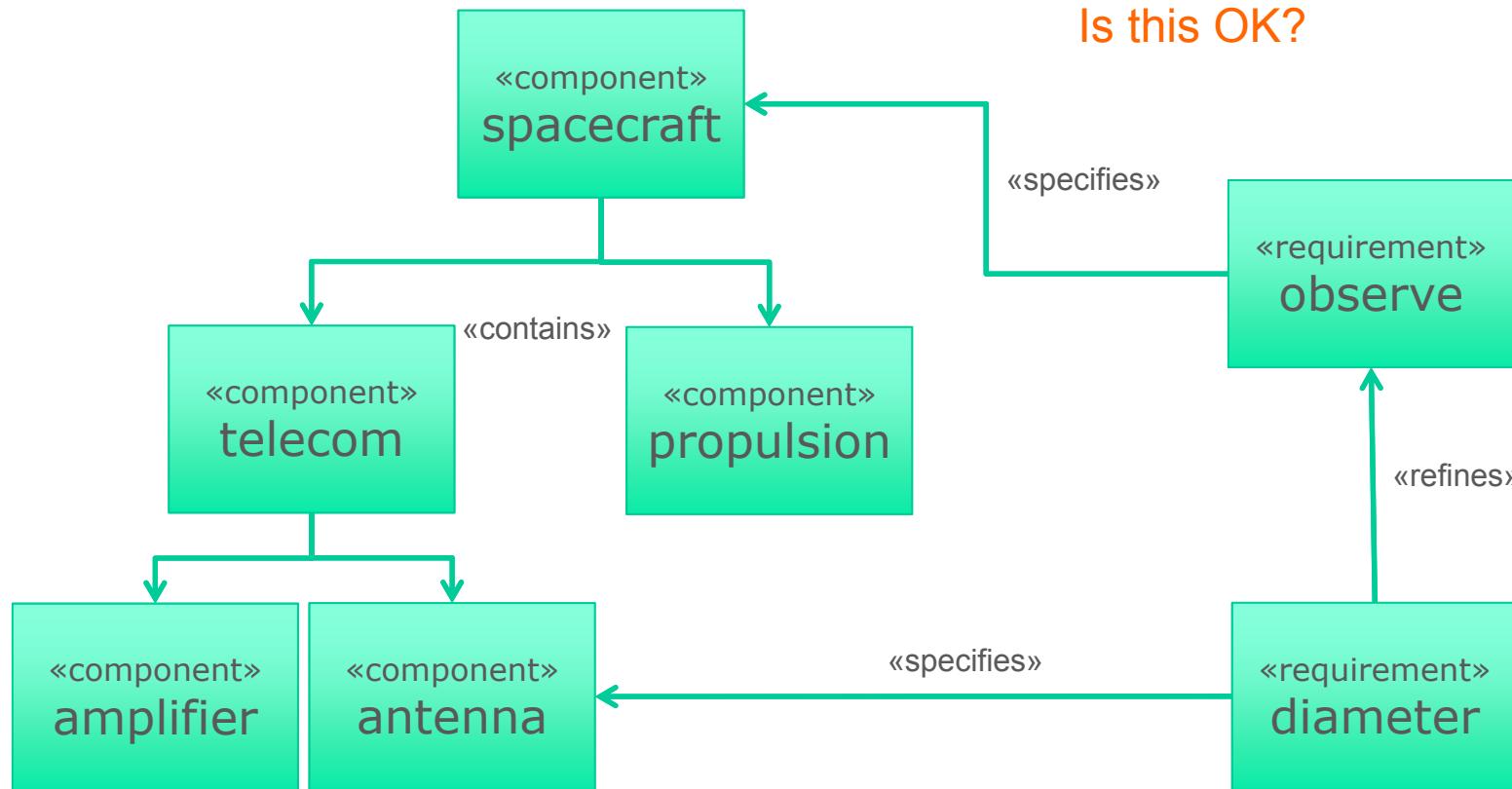
Rule: CBE mass  $m_e$  of any component with parts is the sum of  $m_e$  of its parts

Policy:  $m_e < m_a$  for every component

$m_e$ : estimated mass  
 $m_a$ : allocated mass



# Reasoning About Design



Probably not. Requirements shouldn't jump component levels.

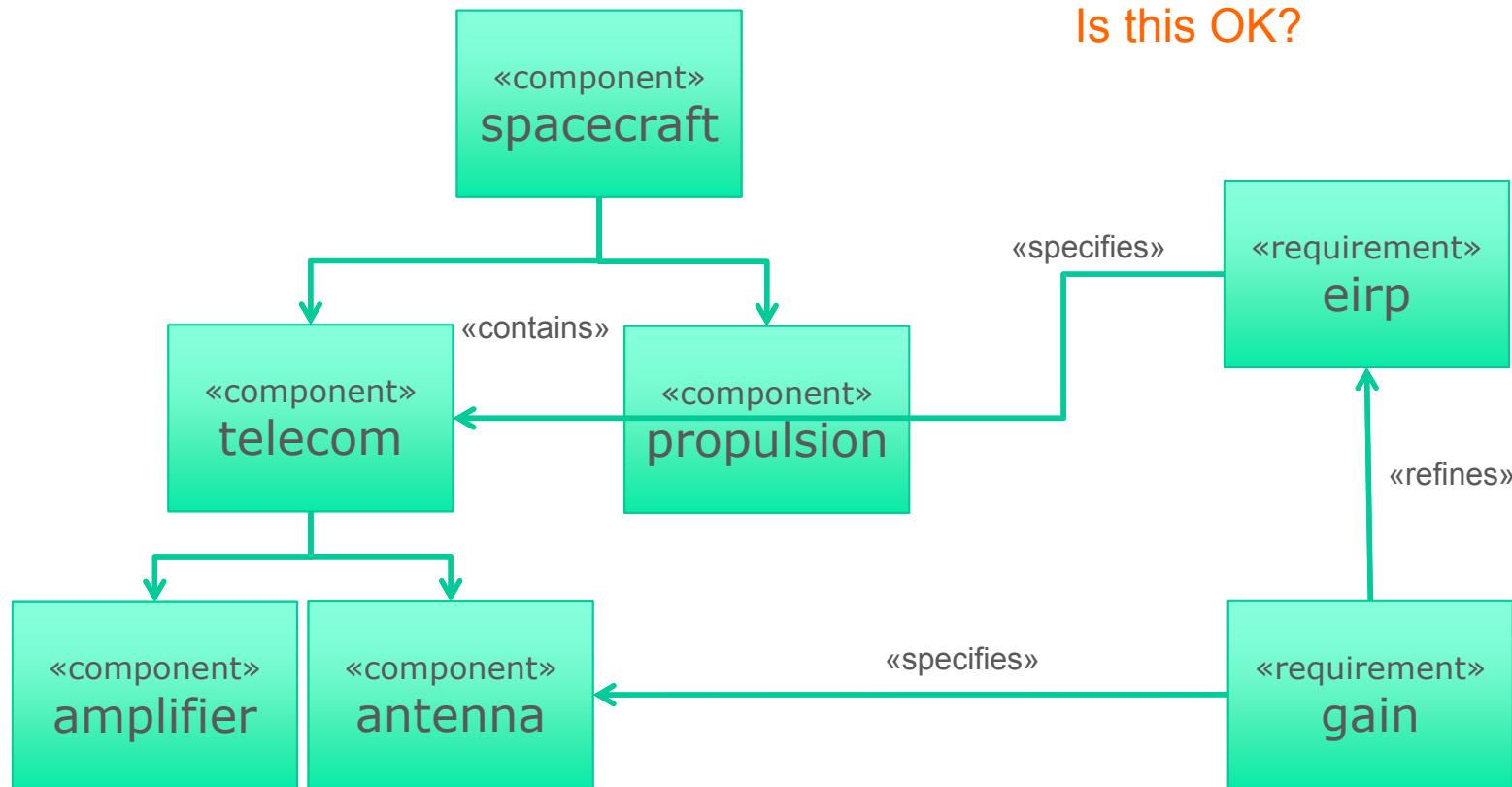


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Reasoning About Design

31 Systems + Software



Is this OK?

Yes. This is a common pattern.

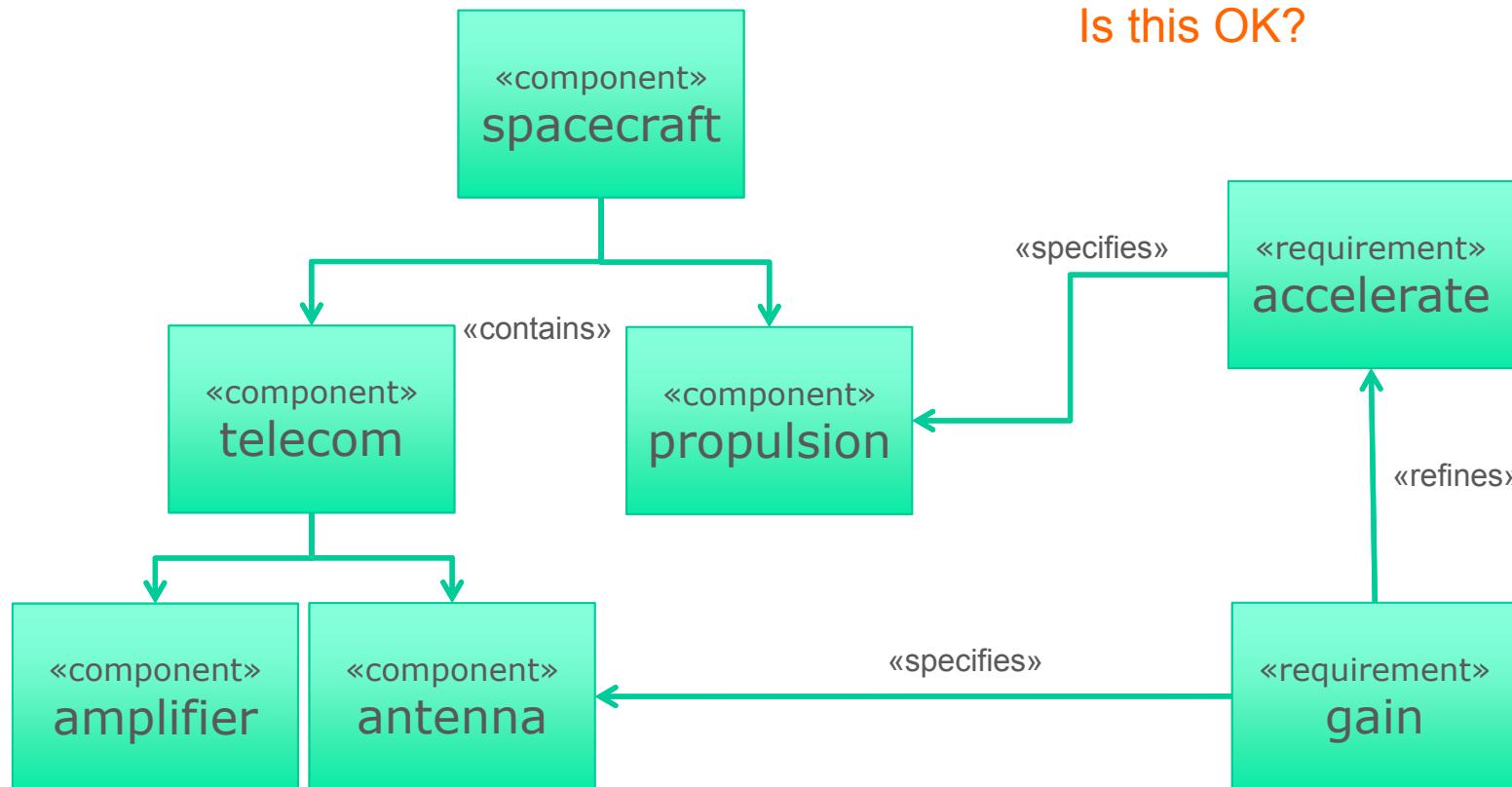


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Reasoning About Design

31 Systems + Software



No. Requirement flowdown should be consistent with product decomposition.

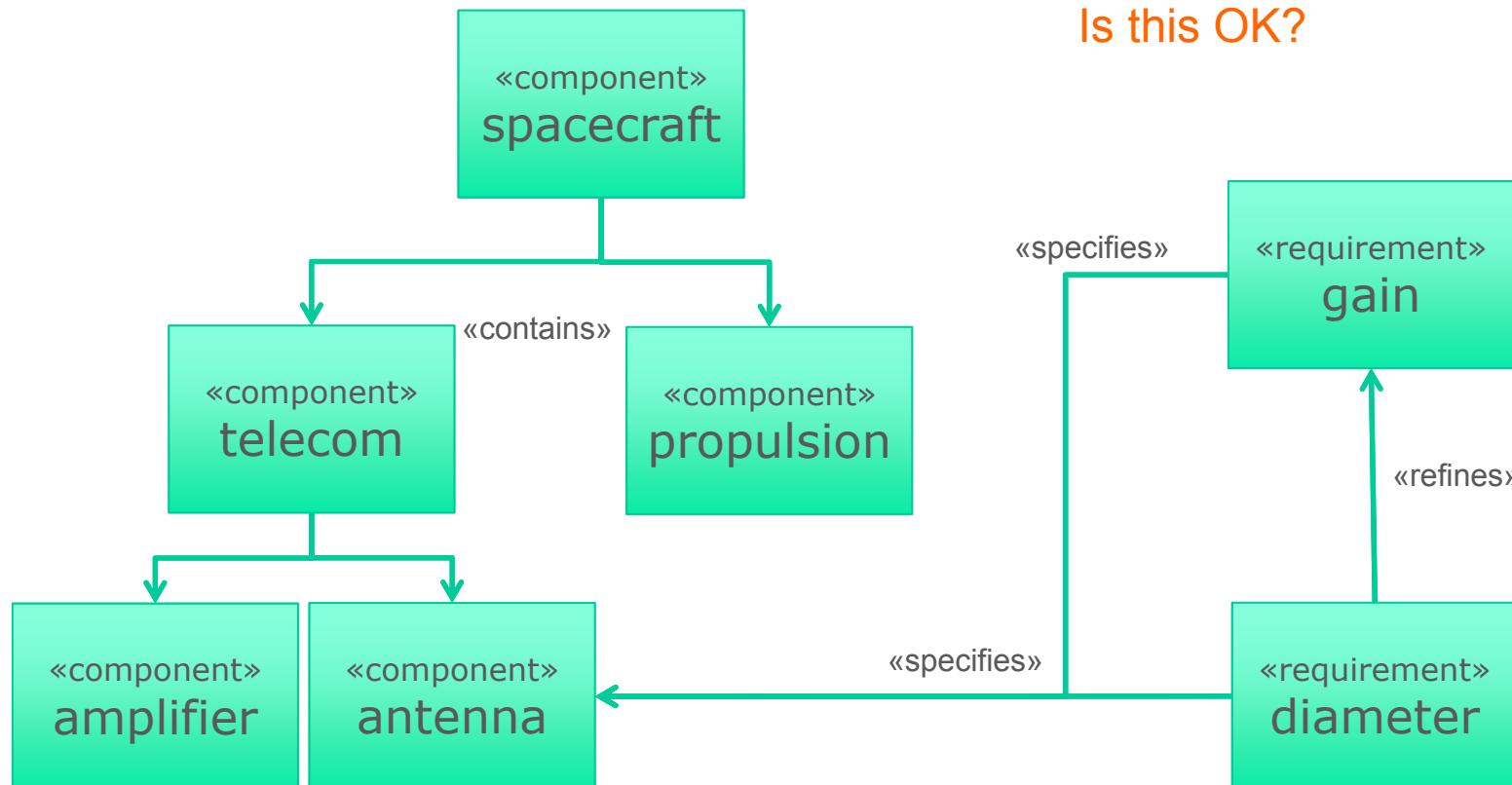


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Reasoning About Design

31 Systems + Software



Yes. Sometimes you decompose at the same level for clarity.



National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Some Objectives of Modeling

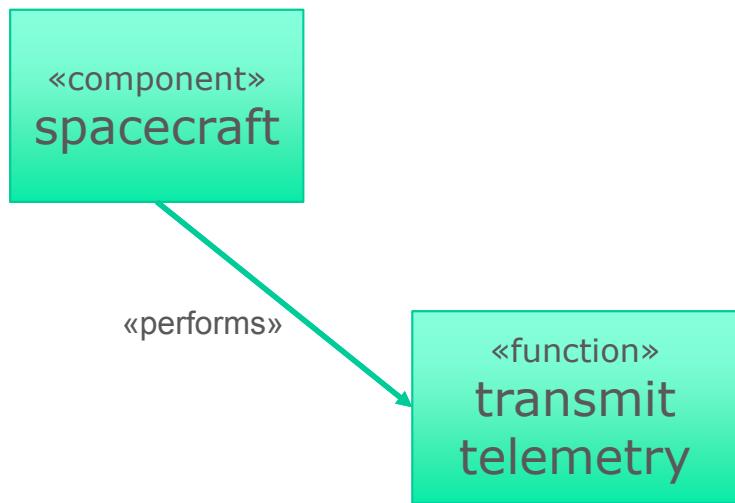
31 Systems + Software

- **To describe a design in durable form**
  - You can use almost anything for that
- **To communicate a design to a set of stakeholders**
  - Now you need (at least) a common notation and familiar presentation idioms
  - Standards (e.g., SysML) cover most of that
- **To organize and relate analyses of a design**
  - This is, in general, a much harder problem
  - You have to make sure that every element that could affect an analysis is present, properly identified, and consistently related to appropriate other elements
    - This is largely outside the scope of SysML, except to provide extension mechanisms that allow you to define the rules
  - You also need software to reason about your models
    - This is also outside the scope of SysML, but some tools do
  - Analysis operates on facts



# Presentations and Facts

## Presentation



**SysML is (among other things) a presentation standard**

## Facts

- ***spacecraft* is a «component»**
- ***transmit telemetry* is a «function»**
- ***spacecraft* «performs» *transmit telemetry***

**We need other standards for our facts**



National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Facts and Ontologies

31 Systems + Software

- The field that deals with facts and reasoning is *logic*
- The subset of logic that deals with facts and their meaning is *ontology*
- Ontologies contain *axioms*:
  - Definitions of concepts and their specializations
    - e.g., a *Spacecraft* is a *Flight Component*, which is a *Component*
    - These are sometimes called *classes*
  - Definitions of attributes of individuals of a class
    - e.g., *mass* is a property of *Flight Component*
    - These are sometimes called *data properties*
  - Definitions of relationships among individuals
    - e.g., a *Component* performs a *Function*
    - These are sometimes called *object properties*
  - Restrictions
    - e.g., a *Function* is *PerformedBy* at most one *Component*
  - Facts about individuals using these concepts and properties



National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Why Do We Care about Ontology?

31 Systems + Software

- **There is a well-developed body of theory that can**
  - **help us avoid undecidable questions**
    - i.e., not solvable in principle
  - **help us avoid intractable questions**
    - i.e., solvable in principle but not in practice
- **There is a body of tools that can**
  - **help us edit our ontologies**
  - **validate our ontologies**
    - i.e., tell us if they're well-formed, consistent, and satisfiable
  - **compute inferences**
    - i.e., *JEO* is a *Spacecraft* and *Spacecraft* is a *Component* implies *JEO* is a *Component*
    - these are sometimes called *entailments*
  - **answer a large class of questions about facts**
    - i.e., What *Components* perform a *Function* that *sends* or *receives* the particular *Message*?



# Ontologies as Integrating Standards

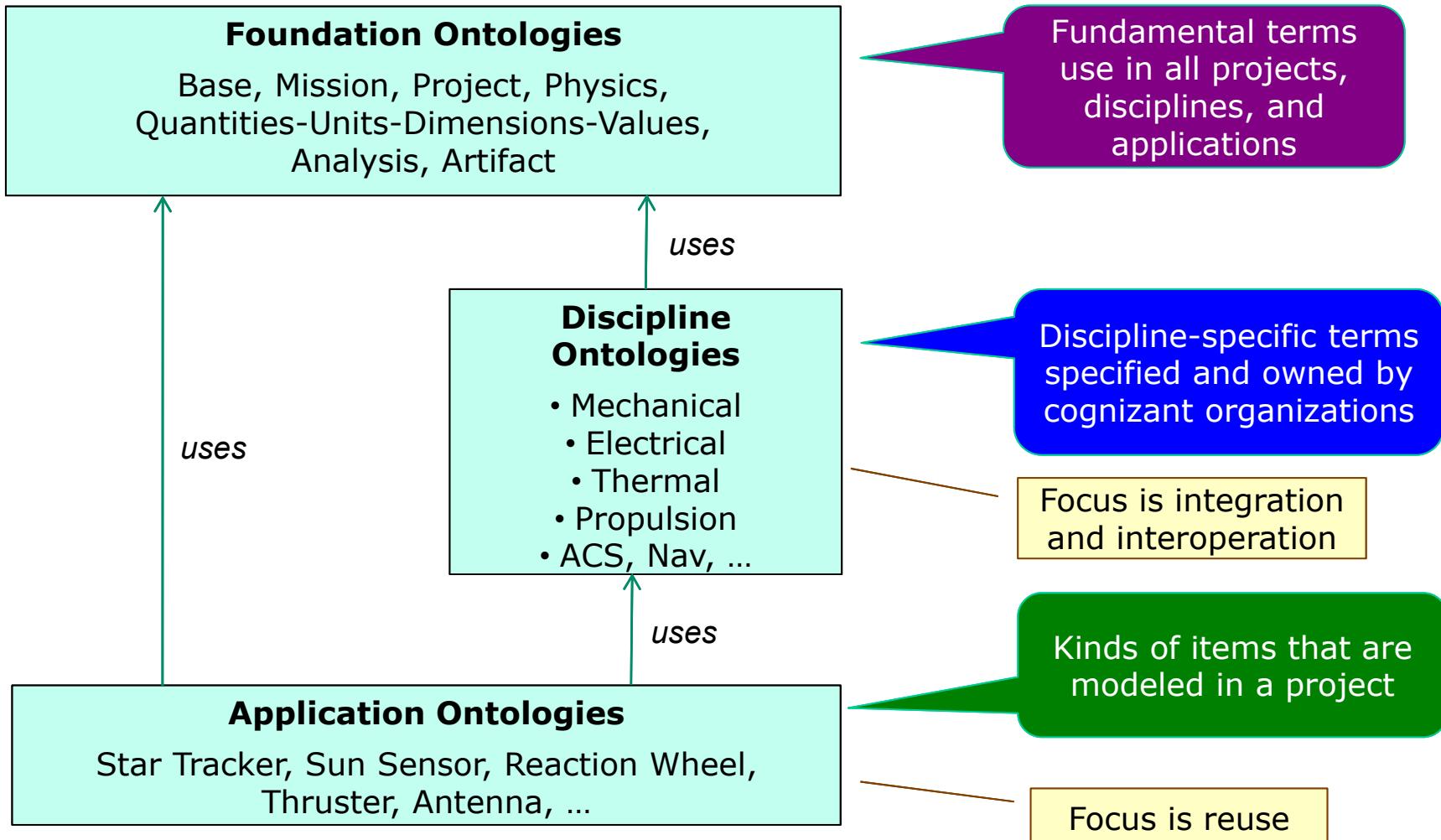
31 Systems + Software

- **We use a lot of discipline-specific tools and terminology in space flight systems engineering**
  - e.g., trajectory synthesis, radiation effects modeling
  - SysML supports the broad discipline of systems engineering, but we need a unifying vocabulary that can relate these disciplines to each other
- **This problem is not unique to space flight (nor to systems engineering)**
  - Lots of people have been working on it for years.
- **There is a set of international (W3C) standards for defining and using ontologies**
  - All related to the Web Ontology Language (OWL)
- **We're building OWL ontologies for disciplines of interest**



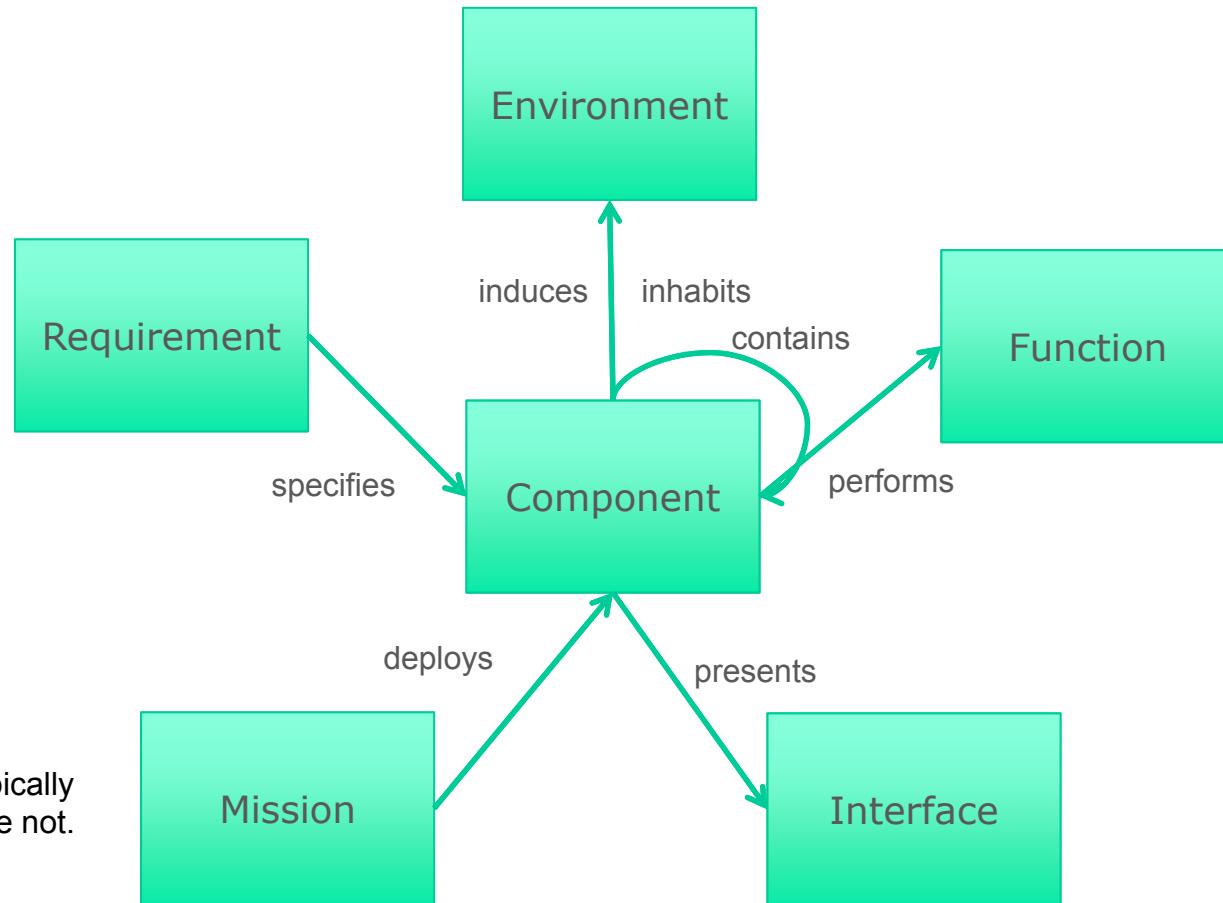
# JPL IMCE Ontology Organization

31 Systems + Software





# Foundation Ontology Example



Concepts in OWL are typically capitalized, properties are not.

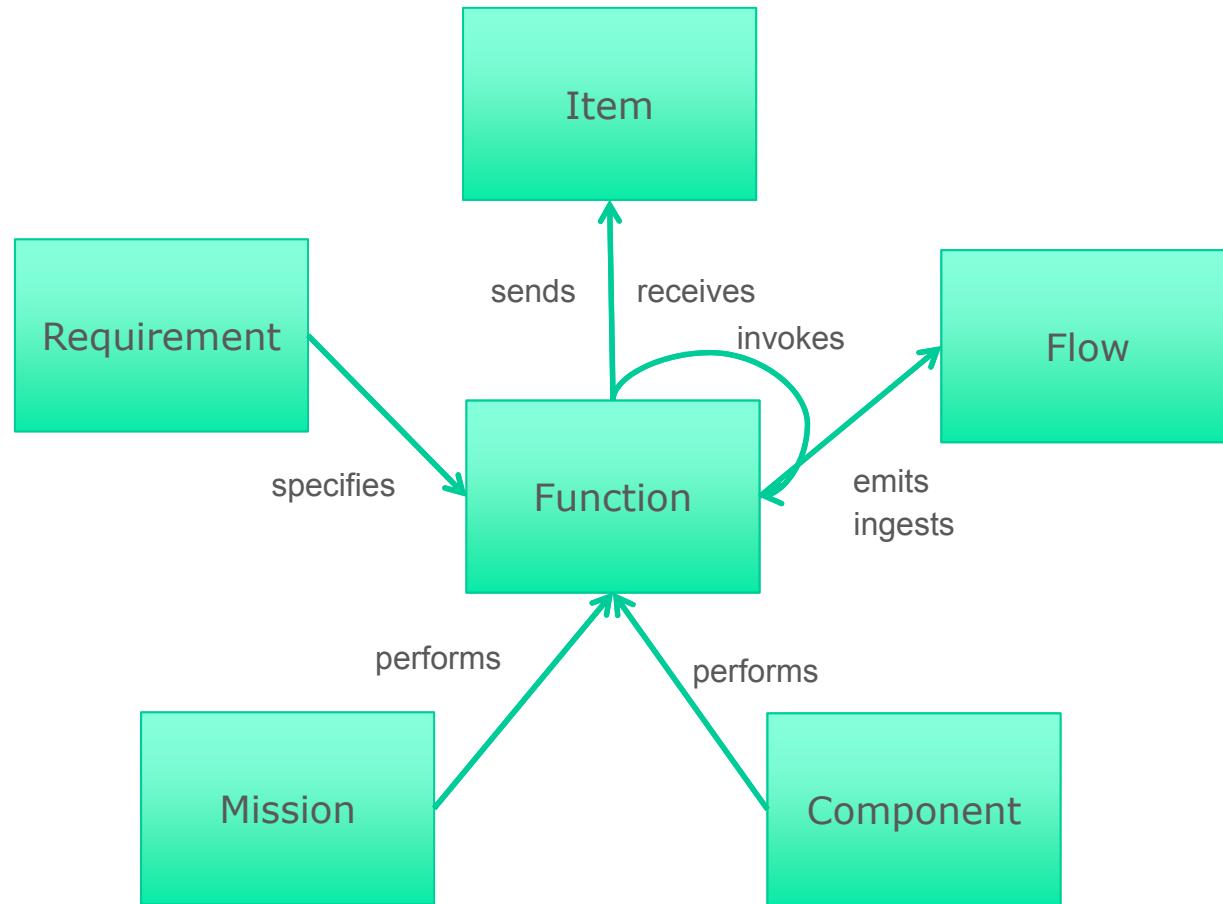


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Foundation Ontology Example

31 Systems + Software



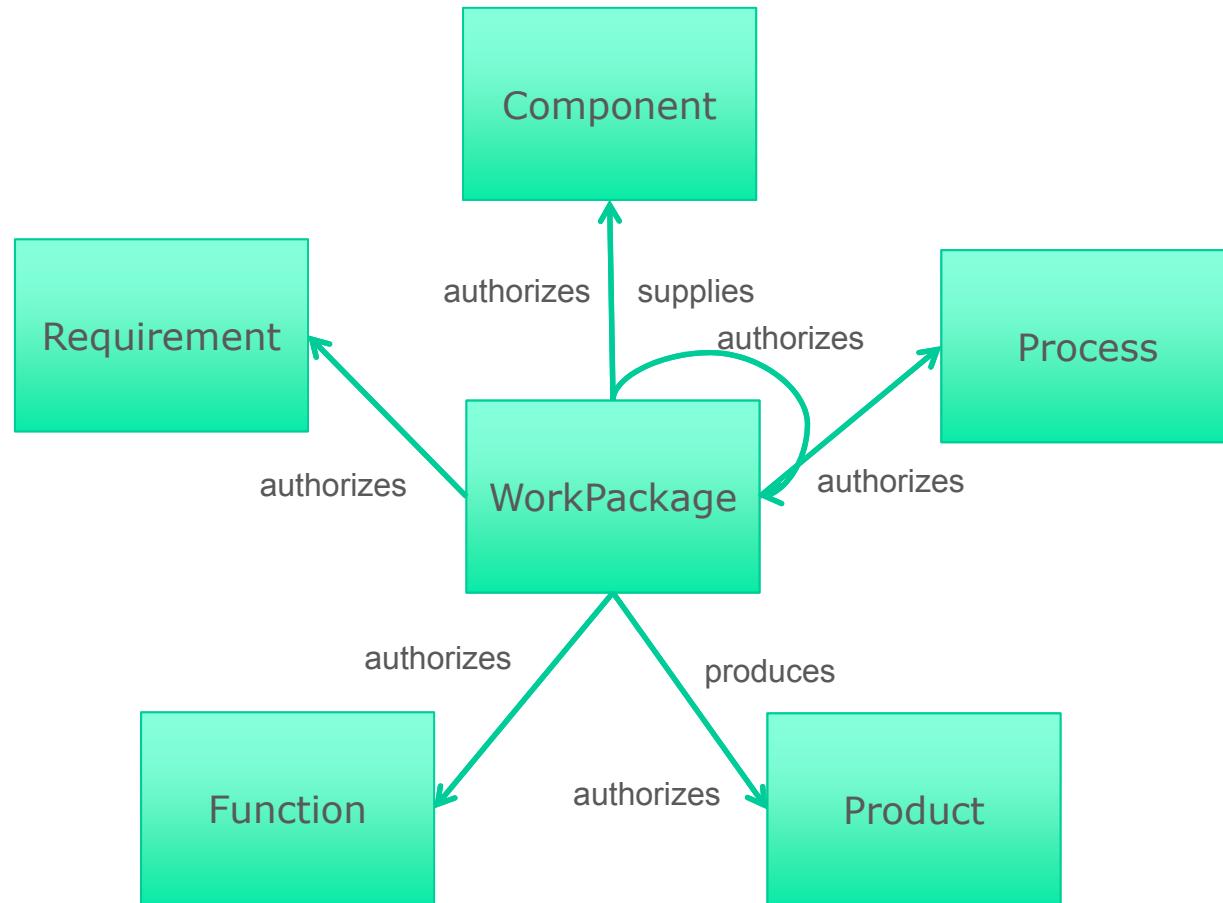


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Foundation Ontology Example

31 Systems + Software



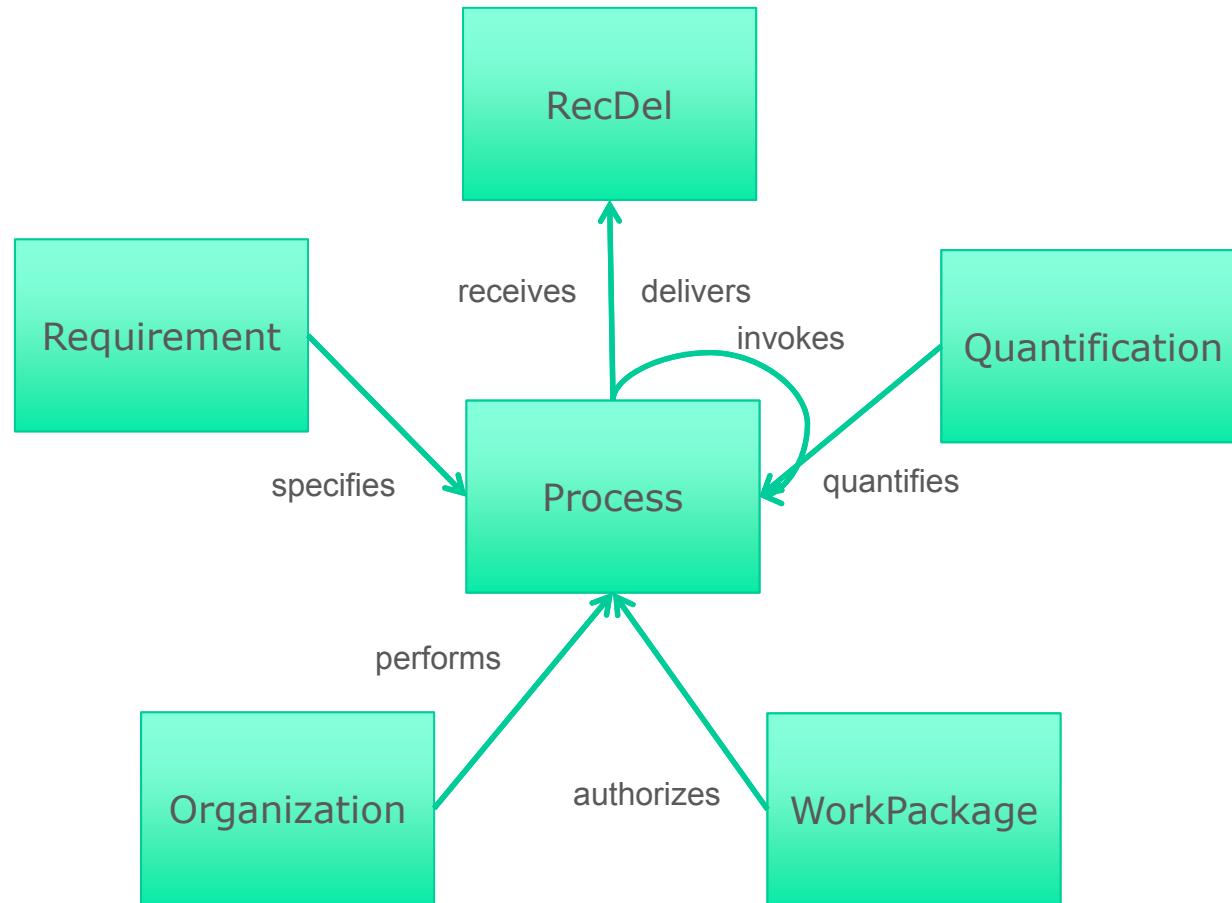


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Foundation Ontology Example

31 Systems + Software





National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Ontologies and SysML

31 Systems + Software

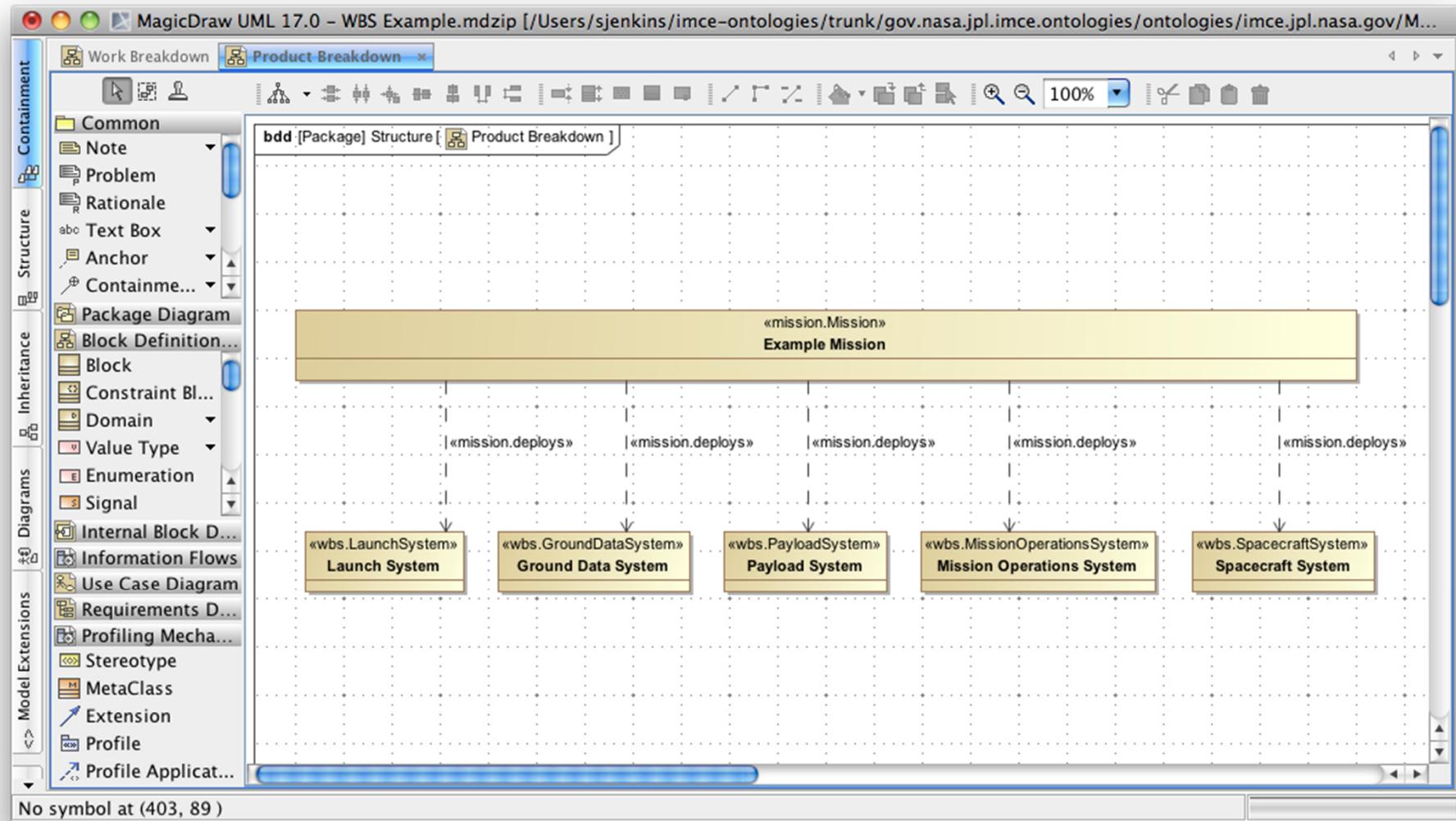
- **SysML contains an extension mechanism for user-defined types and properties**
- **A collection of these extensions is called a *profile***
- **We generate profiles by transforming ontologies**
- **This ensures that**
  - **OWL concept and property definitions are consistent with SysML stereotypes**
  - **SysML “instance” models can be translated to corresponding OWL models for reasoning and analysis**
- **OWL is well-suited to building long-term, tool-neutral archives of project and mission designs**



National Aeronautics and  
Space Administration  
  
Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Example of SysML Profile Application

## 31 Systems + Software



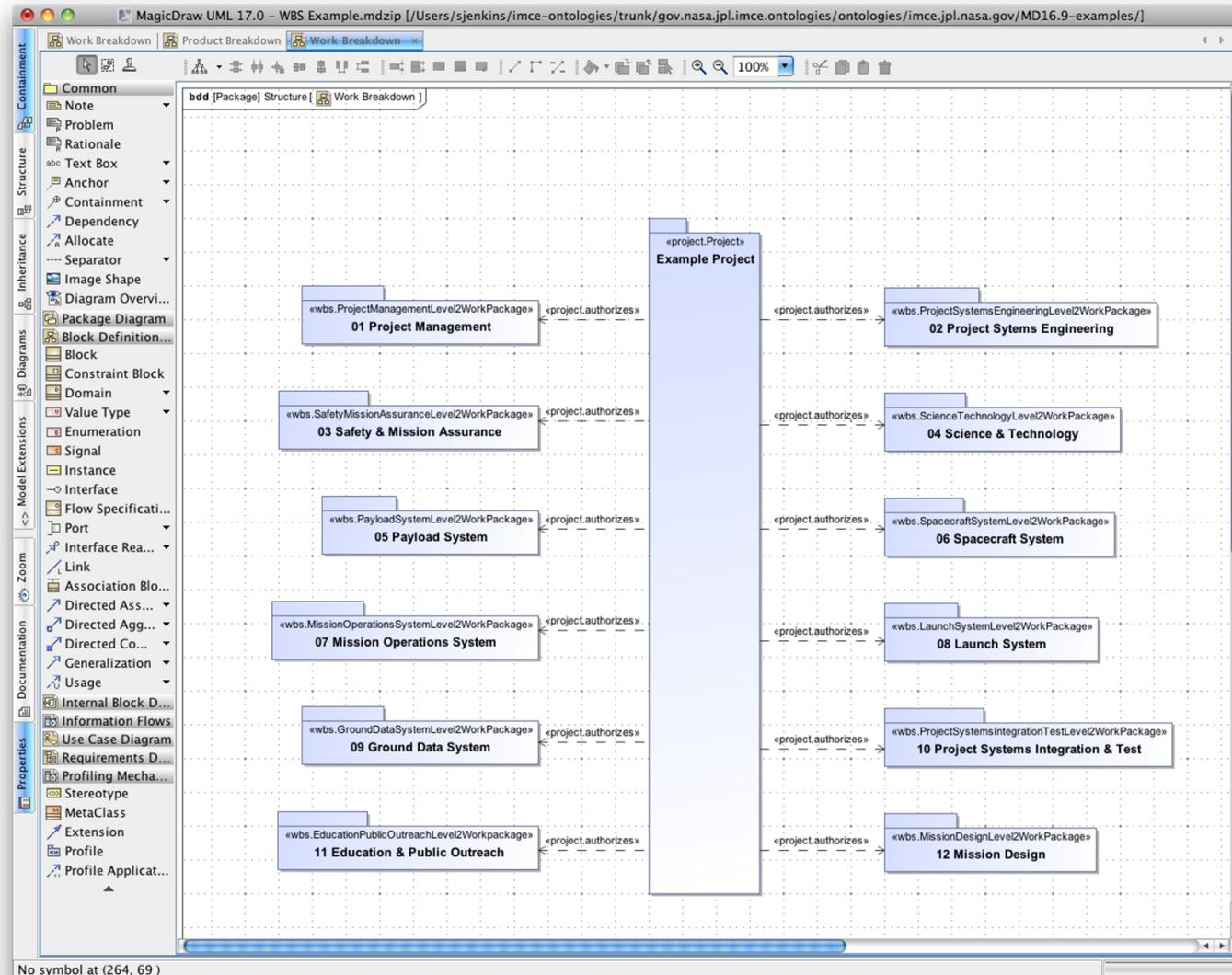


National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

# Example of SysML Profile Application

## 31 Systems + Software





National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

## Closing Thoughts

31 Systems + Software

- Try to keep in mind the idea of classifying things and their relationships with types that are meaningful for space flight in general and JEO in particular
- These classifications are a natural extension of the basic vocabulary of SysML
- They enable the reasoning that is essential for an undertaking of the complexity of space flight